Matematički fakultet, Univerzitet u Beogradu

Seminarski rad

Statistički softver 3

Mentor:
Danijel Subotić

Student:
Aleksandra Radulović

Jun, 2020.

Sadržaj

Ostalo

Zadatak 1.

Za slike koje se nalaze u folderu prvi zadatak potrebno je:

a) Odrediti ugao rotacije geometrijske figure u odnosu na x-osu.

b) Izrotirati figuru, t.d. donja stranica bude paralelna sa x-osom.

v) Sačuvati novodobijenu sliku. Figure su samo pravougaonici proizvoljnih dimenzija.

Rešenje:

Pošto radimo sa slikama, učitavamo imager paket.
Zatim funkcijom setwd() postavljamo folder prvi_zadatak kao trenutni radni direktorijum.
Potom učitavamo slike u listu.

```
library(imager)
setwd("C:\\Users\\Radulovic\\Desktop\\prvi_zadatak")
files <- list.files(path=getwd(),all.files=T, full.names=F, no.. = T)
slike <- lapply(files, load.image)
```

Sada ćemo napraviti funkciju koja pronalazi pravougaonik na slici.

Pravougaonici se na svakoj slici od pozadine razlikuju po boji. Možemo da identifikujemo piksele koji pripadaju pozadini, i one koji pripadaju pravougaoniku tako što ćemo naći srednju vrednost neke boje, pa na osnovu toga da li je vrednost boje odgovarajućeg piksela veća ili manja od srednje vrednosti, prepoznajemo da li pripada pravougaoniku ili ne.

Koristeći ovo možemo da pronađemo krajnje granične tačke figure sa svake strane(levo,desno, gore i dole), i to su temena pravougaonika. Kada imamo temena, lako nalazimo i centar figure, koji nam treba za rotaciju.

U zadatku se traži i ugao rotacije. Njega možemo naći na sledeći način.

Ako posmatramo pravougli trougao sa temenima $(x1, y1), (x2, y2), (x2, y1)$, tangens ugla kod temena $(x1, x2)$ je odnos naspramne i nalegle katete, odnosno $tan(y/x)$, gde je $y = y2 - y1, x = x2 - x1$.

Naš ugao rotacije će biti arkus tangens od tog ugla, $arctan(y/x)$. Pošto će rezultat biti u radijanima, potrebno je još da ga pretvorimo u stepene, odnosno pomnožimo sa $180/\pi$.

```
funkcija=function(slika,b){ # b je kanal boje

  slika=renorm(slika)#Primenom funkcije "renorm()" trebalo bi da se dobije
  #normalizovani oblike svake slike i svi objekti koju se drugacije zapisu
ju u R-u,
  #ali izgledaju isto bi trebalo posle primene funkcije renorm() postanu (
skoro) isti.
  matrica=slika[,,1,b]
  dimenzije_matrice=dim(matrica)
  I=(matrica>mean(matrica))  #ovo će nam biti indikator tačaka različitih
od pozadine
```

```
  #postavljamo početne vrednosti za temena
  krajnja_leva_tacka=dimenzije_matrice
  krajnja_desna_tacka=c(0,0)
  krajnja_gornja_tacka=dimenzije_matrice
  krajnja_donja_tacka=c(0,0)

  #for petljom prolazimo kroz matricu da bismo našli temena
  #prvo identifikujemo tačke različite od pozadine,
  #onda se kroz njih krećemo po širini i visini, tačka koja ima najmanju š
irinu je krajnja leva tačka,
  #ona sa najvećom širinom je krajnja desna,
  #krajnja gornja tačka ima najmanju vrednost promenljive visina
  #krajnja donja najveću vrednost te promenljive
  for(sirina in 1:dimenzije_matrice[1] ){

    for(visina in 1:dimenzije_matrice[2]){

       if(I[1,1]!=I[sirina,visina]){

          if(krajnja_leva_tacka[1]>sirina) {krajnja_leva_tacka=c(sirina,vi
sina)}


         if(krajnja_desna_tacka[1]<sirina) {krajnja_desna_tacka=c(sirina,v
isina)}

         if(krajnja_gornja_tacka[2]>visina){krajnja_gornja_tacka=c(sirina,
visina)}

         if(krajnja_donja_tacka[2]<visina){krajnja_donja_tacka=c(sirina,vi
sina)}

      }
   }
}

  centar=(krajnja_gornja_tacka+krajnja_donja_tacka+krajnja_leva_tacka+kraj
nja_desna_tacka)/4

  #nalazimo ugao na način objašnjen u tekstu
  xy=abs(krajnja_donja_tacka-krajnja_desna_tacka)
  trazeni_ugao=atan2(xy[2],xy[1])*((180)/pi)
  return(c(trazeni_ugao,centar))}
```

Sada nam treba funkcija koja će rotirati pravougaonike za taj ugao.Koristimo funkciju imrotate().

```
#funkcija koja rotira sliku za dati ugao, i zatim prikazuje sliku:
rotacija_za_dati_ugao<-function(slika,ugao,cx,cy){
  rotirana_slika=imrotate(slika,ugao,round(cx),round(cy))
  plot(rotirana_slika)
  return(rotirana_slika)
}
```

Sada možemo da primenimo ovu funkciju na naše slike. Ostalo je samo da napravimo folder u kome se slike čuvaju:

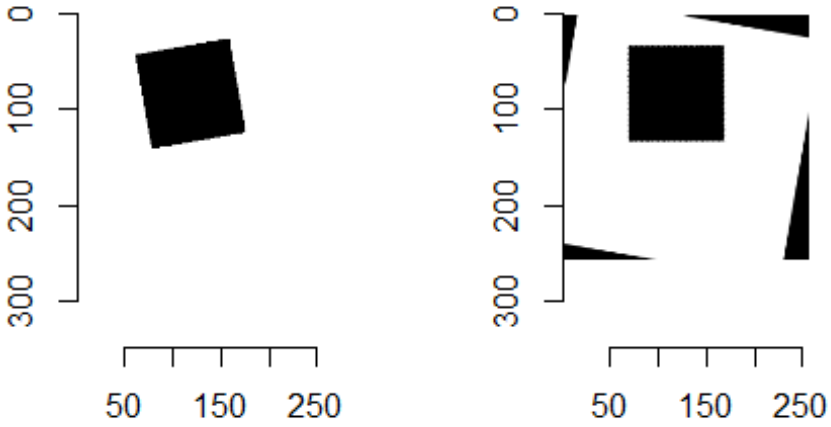
```
dir.create(file.path(getwd(),"slike"))

## prva slika
par(mfrow=c(1,2))
slika1=slike[[1]]
plot(slika1)

# a)
#traženi ugao rotacije:
funkcija(slika1,2)[1]

## [1] 9.841132

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika1=rotacija_za_dati_ugao(slika1,funkcija(slika1,2)[1],funkcij
a(slika1,2)[2],funkcija(slika1,2)[3])
```



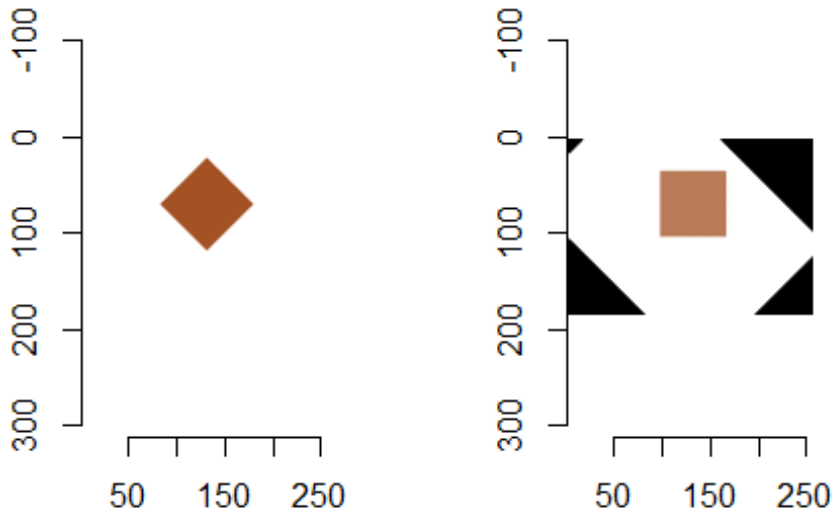

```
#v)čuvamo sliku
save.image(rotirana_slika1,paste("slike\\",strsplit(files[1],".",fixed = T
)[[1]][1],".png",sep=""))

## druga slika
slika2=slike[[2]]
plot(slika2)

#traženi ugao rotacije:
funkcija(slika2,2)[1]
```

```
## [1] 45

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika2=rotacija_za_dati_ugao(slika2,funkcija(slika2,2)[1],funkcij
a(slika2,2)[2],funkcija(slika2,2)[3])
```



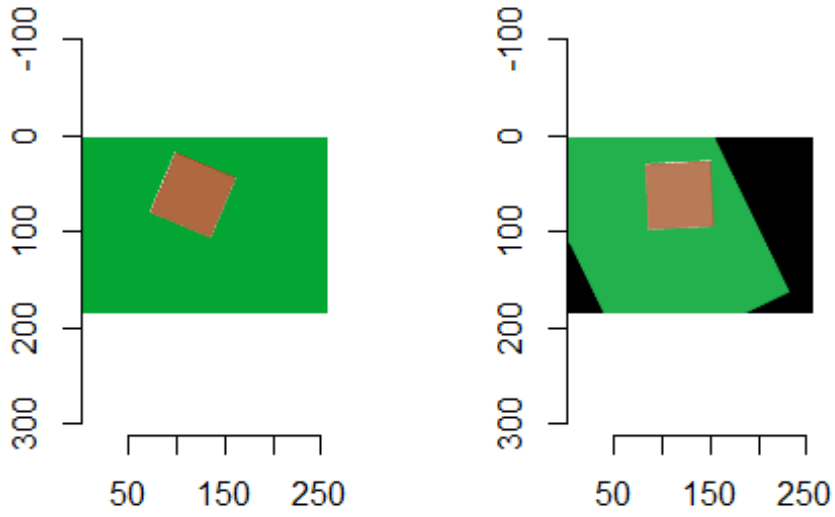

```
#v)čuvamo sliku
save.image(rotirana_slika2,paste("slike\\",strsplit(files[2],".",fixed = T
)[[1]][1],".png",sep=""))


## treća slika
slika3=slike[[3]]
plot(slika3)

#traženi ugao rotacije:
funkcija(slika3,2)[1]

## [1] 64.17901

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika3=rotacija_za_dati_ugao(slika3,funkcija(slika3,2)[1],funkcij
a(slika3,2)[2],funkcija(slika3,2)[3])
```



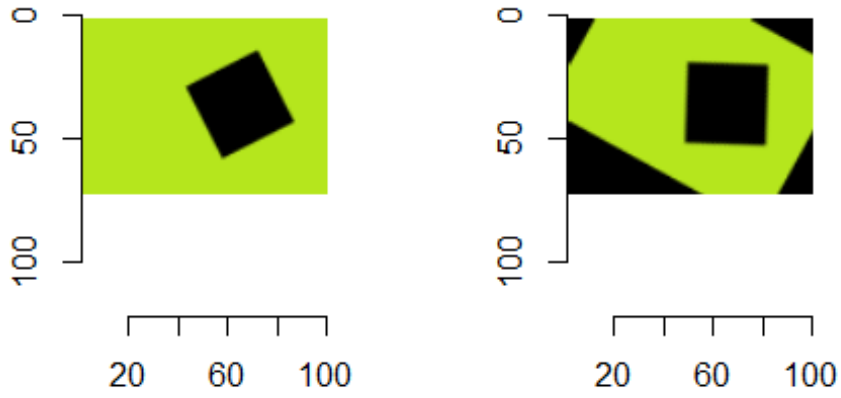

```
#v)čuvamo sliku
save.image(rotirana_slika3,paste("slike\\",strsplit(files[3],".",fixed = T
)[[1]][1],".png",sep=""))

## četvrta slika
slika4=slike[[4]]
plot(slika4)

#traženi ugao rotacije:
funkcija(slika4,2)[1]

## [1] 28.88658

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika4=rotacija_za_dati_ugao(slika4,funkcija(slika4,2)[1],funkcij
a(slika4,2)[2],funkcija(slika4,2)[3])
```



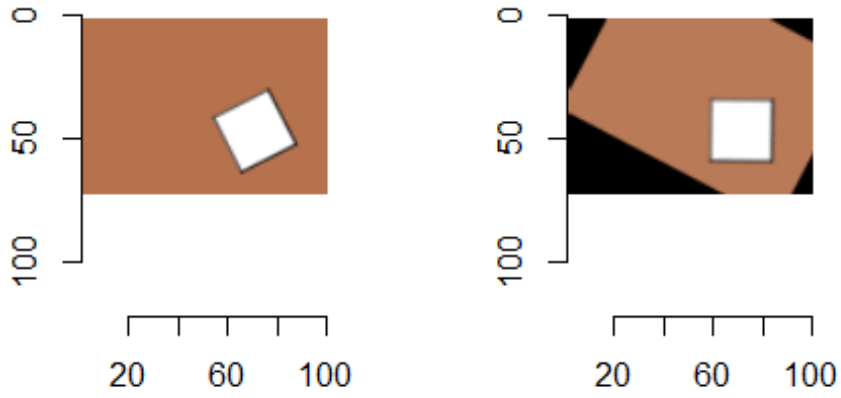

```
#v)čuvamo sliku
save.image(rotirana_slika4,paste("slike\\",strsplit(files[4],".",fixed = T
)[[1]][1],".png",sep=""))


## peta slika
slika5=slike[[5]]
plot(slika5)

#traženi ugao rotacije:
funkcija(slika5,2)[1]

## [1] 27.64598

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika5=rotacija_za_dati_ugao(slika5,funkcija(slika5,2)[1],funkcij
a(slika5,2)[2],funkcija(slika5,2)[3])
```



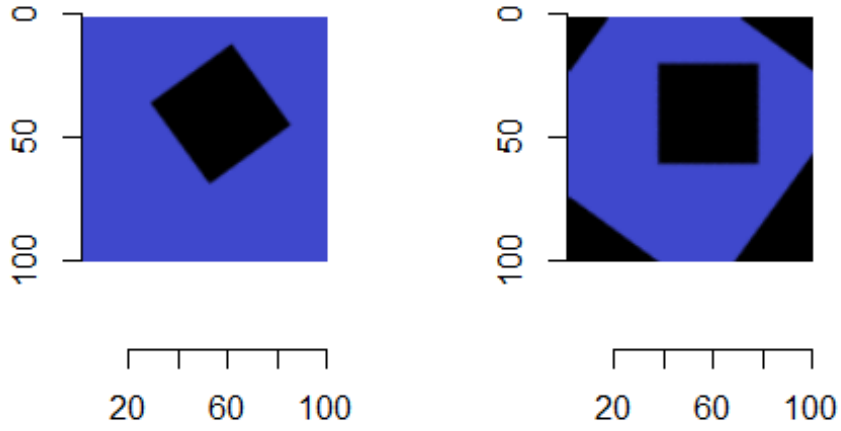

```
#v)čuvamo sliku
save.image(rotirana_slika5,paste("slike\\",strsplit(files[5],".",fixed = T
)[[1]][1],".png",sep=""))

## šesta slika
slika6=slike[[6]]
plot(slika6)

#traženi ugao rotacije:
funkcija(slika6,2)[1]

## [1] 36.02737

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika6=rotacija_za_dati_ugao(slika6,funkcija(slika6,2)[1],funkcij
a(slika6,2)[2],funkcija(slika6,2)[3])
```



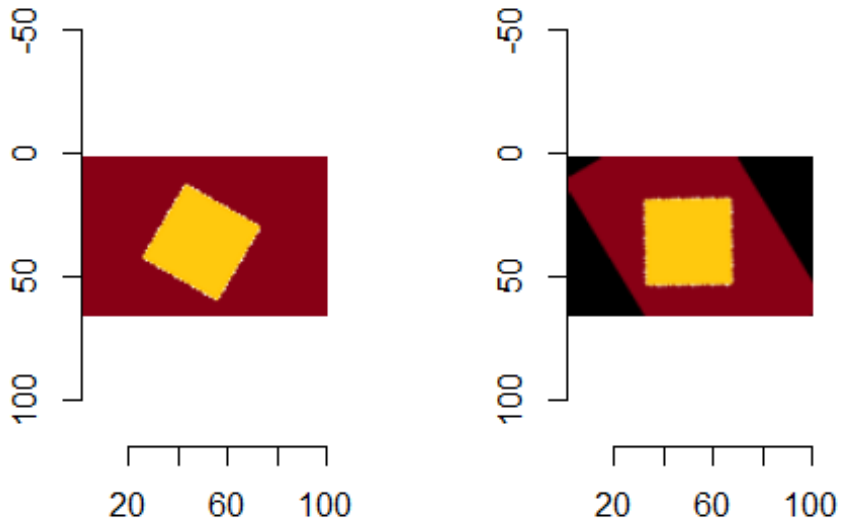

```
#v)čuvamo sliku
save.image(rotirana_slika6,paste("slike\\",strsplit(files[6],".",fixed = T
)[[1]][1],".png",sep=""))

## sedma slika
slika7=slike[[7]]
plot(slika7)

#traženi ugao rotacije:
funkcija(slika7,2)[1]

## [1] 59.03624

#b)rotiramo sliku za dati ugao, pa će biti paralelna sa x osom
rotirana_slika7=rotacija_za_dati_ugao(slika7,funkcija(slika7,2)[1],funkcij
a(slika7,2)[2],funkcija(slika7,2)[3])
```




```
#v)čuvamo sliku
save.image(rotirana_slika7,paste("slike\\",strsplit(files[7],".",fixed = T
)[[1]][1],".png",sep=""))
```

Zadatak 2.

Napisati program pomoću kojeg se simlira igra iks oks. Osposobiti da računar bude protivnik koji svoje poteze određuje pomoću Monte Karlo simulacije.

Rešenje:

Učitavamo "lava paket", jer se u njemu nalazi revdiag() funkcija, koju ćemo koristiti da proverimo elemente na dijagonalama naše table.

```
#install.packages("lava")
library(lava)
```

Prvo pravimo funkciju "napravi potez" koja koristeći Monte Karlo metodu određuje koji je najbolji potez koji igrač koji je trenutno na potezu može da povuče.

Ideja je da u svakom koraku, računar dobije informaciju u koje polje igrač koji je na potezu treba da stavi svoj simbol na sledeći način:

For petljom program prolazi kroz sve moguće pozicije. Za svaku od njih, postavi simbol na tu poziciju, a onda N puta popuni preostale pozicije preostalim brojem x i o simbola, i beleži koliko puta je ostvarien uslov za pobedu, odnosno 3 odgovarajuća simbola u vrsti,koloni ili dijagonali. Broj pobeda za svaku poziciju se čuva u matrici, i na kraju za svaku poziciju broj_pobeda/N uzimamo kao verovatnoću da pobedi u igri ako za sledeći potez izabere baš

tu poziciju. Računar bira za koju poziciju je ta verovatnoća najveća, i u nju bira da upiše simbol.

Pretpostavljamo da prvi igrač uvek počinje sa x simbolom.

Argumenti funkcije su N - broj MK simulacija, matrica koja predstavlja tablu za iks-oks i id_prvi, koji je uzima vrednost true ako je na redu prvi igrač, false u suprotnom.

```
napravi_potez<-function(N,matrica,id_prvi){


vektorizacija=as.vector(matrica) #matricu pretvaramo u vektor

  M=matrix(rep(0,9),ncol=3) #matrica u kojoj ćemo čuvati verovatnoće pozic
ija



  moguce_pozicije=which(vektorizacija==' ') #pozicije koje su još neotvore
ne

#preostali broj x i o koji mogu da se iskoriste
  preostalih_x=5-sum(vektorizacija=='X')
  preostalih_o=4-sum(vektorizacija=='O')


  #ako je na redu prvi igrač, on će iskoristiti jedan X, pa smanjujemo br
preostalih ikseva
  if(id_prvi==1){
    preostali_xo=c(rep('X',(preostalih_x)-1),rep('O',preostalih_o))
  }


   #analogno za drugog igrača i preostale o simbole
   else{
    preostali_xo=c(rep('X',preostalih_x),rep('O',(preostalih_o-1)))
  }
 #prolazimo kroz sve moguće pozicije for petljom
   for(i in moguce_pozicije){
    moguce=moguce_pozicije[moguce_pozicije!=i]

    for(k in 1:N){

      if(length(moguce)>1){
        x=sample(moguce,length(moguce),replace=F)}

       else
        x=moguce


      y=preostali_xo

      for(l in 1:length(moguce)){
```

```
        vektorizacija[x[l]]=y[l]

      }
      vektorizacija[i]=ifelse(id_prvi,'X','O')
      matr=vektorizacija
      dim(matr)=c(3,3)


      #proveravamo da li je prvi igrac na potezu i na tabli imamo 3 x simb
ola u vrsti, koloni ili na nekoj od dijagonala
      if(id_prvi==1 & (all(matr[1,]=='X') | all(matr[2,]=='X') | all(matr[
3,]=='X') | all(matr[,1]=='X') | all(matr[,2]=='X') | all(matr[,3]=='X') )
){
        M[i]=M[i]+1
      }
      else if(id_prvi==1 & (all(diag(matr)=='X') | all(revdiag(matr)=='X')
)){
        M[i]=M[i]+1
      }


      #analogno za drugog igrača
      else if(id_prvi==0 & (all(matr[1,]=='O') | all(matr[2,]=='O') | all(
matr[3,]=='O') | all(matr[,1]=='O') | all(matr[,2]=='O') | all(matr[,3]=='
O') )){
        M[i]=M[i]+1
      }
      else if(id_prvi==0 & (all(diag(matr)=='O') | all(revdiag(matr)=='O')
)){
        M[i]=M[i]+1}

      else{

      }  }
  }

  M=M/N
  return(M)} #vraćamo matricu verovatnoća
```

Zatim funkcija koja pušta simulaciju

```
pusti_simulaciju<-function(){
matrica=matrix(rep(" ",9),ncol=3)

#najviše može biti 9 koraka
for(i in 1:9){

  message("pocinje ",i, ".  potez")
  id_prvi=i%%2
  m=napravi_potez(N=i*3000,matrica=matrica,id_prvi)
  verovatnoca=m[which.max(m)]
  pozicija=which(verovatnoca==m)
  matrica[pozicija[1]]=ifelse(id_prvi,'X','O')
  print(matrica)
```

```
  #igra je gotova kada je neki od igrača popunio 3 polja u vrsti,koloni, i
li jednoj od dijagonala

  if((id_prvi==1) & (all(matrica[1,]=='X') | all(matrica[2,]=='X') | all(m
atrica[3,]=='X') | all(matrica[,1]=='X') | all(matrica[,2]=='X') | all(mat
rica[,3]=='X') | all(diag(matrica)=='X') | all(revdiag(matrica)=='X') )){


  print("Pobedio je 1. igrac")
  break}
  else if((id_prvi==0) & (all(matrica[1,]=='O') | all(matrica[2,]=='O') |
all(matrica[3,]=='O') | all(matrica[,1]=='O') | all(matrica[,2]=='O') | al
l(matrica[,3]=='O') | all(diag(matrica)=='O') | all(revdiag(matrica)=='O')
)){
    print("pobedio je 2. igrac")
    break}

    }
}
```

I zatim pokrećemo igru:

```
pusti_simulaciju()

## pocinje 1.  potez

##      [,1] [,2] [,3]
## [1,] " "  " "  " "
## [2,] " "  "X"  " "
## [3,] " "  " "  " "

## pocinje 2.  potez

##      [,1] [,2] [,3]
## [1,] " "  " "  " "
## [2,] " "  "X"  " "
## [3,] "O"  " "  " "

## pocinje 3.  potez

##      [,1] [,2] [,3]
## [1,] " "  " "  " "
## [2,] " "  "X"  "X"
## [3,] "O"  " "  " "

## pocinje 4.  potez

##      [,1] [,2] [,3]
## [1,] "O"  " "  " "
## [2,] " "  "X"  "X"
## [3,] "O"  " "  " "

## pocinje 5.  potez
```

```
##      [,1] [,2] [,3]
## [1,] "O"  " "  " "
## [2,] "X"  "X"  "X"
## [3,] "O"  " "  " "
## [1] "Pobedio je 1. igrac"
```

Minolovac

Zadatak 4.

Za slike table Minolovca, koje se nalaze u folderu cetvrti_zadatak_obucavanje napraviti modele (QDA,LDAi Multinomni Logistički) čiji je zadatak da se odredi koji se broj nalazi u svakom od potpoplja. Testirati koliko su dobri traženi modeli na slikama koje su u folderu cetvrtizadatak kontrolni. Detaljno opisati sve iskorišćene funkcije i prediktore, prokomentarisati rezultate, odrediti koji jemodel najbolji. Napomena: Obavezno smisliti i neke principijelno nove prediktore.

Rešenje:

Prvo učitavamo potrebne pakete, imager, MASS i nnet

```
library(imager)

library(MASS)
library(nnet)

#učitavamo slike

setwd("C:\\Users\\Radulovic\\Desktop\\cetvrti_zadatak_obucavanje")
files <- list.files(path=getwd(),all.files=T, full.names=F, no.. = T)
slike <- lapply(files, load.image)
```

Pravimo funkciju koja određuje granice, kao sa časa. Pošto su na nekim slikama odsečene krajnje granice slike, da ne bismo imali problem pri prolasku kroz sva polja, dodajemo veštacki kranje granice, odnosno ako prva granica počinje tek oko 80. piksela, stavićemo da je prvi element vektora koji sadrži granice 1, da ne bismo gubili prvo polje. Analogno za krajnji element vektora.

```
granice<-function(matrica,epsilon,prob,by.row=T){
  if(by.row){
    Mean<-rowMeans}

  else{
    Mean<-colMeans}
  vec<-Mean(matrica<epsilon)>=prob
  granica=which(vec!=F)

  if(granica[1]>79)

    granica=c(1,granica)
  if(granica[length(granica)]<720)
```

```
    granica=c(granica,721)
  return(granica)
}
```

Zatim pravimo funkciju koja sređuje granice:

```
sredjivanje <- function(vektor)
{
  novi_vektor <- c()
  for(i in 1:(length(vektor)-1))
  {
    if(vektor[i+1]-vektor[i]>5){
      novi_vektor <- c(novi_vektor, vektor[i])
    }
  }
  for(j in length(vektor):2){
    if(vektor[j]-vektor[j-1]>5)
  {
      novi_vektor <- c(novi_vektor, vektor[j])
    }
  }
  return(sort(novi_vektor))
}
```

Sada pravim vektore Yk koji predstavljaju kategoričku promenljivu za svaku sliku, a zatim ih sve spajamo u jedan vektor koji će činiti kategoričku promenljivu Y.

```
#vektori sa vrednostima za kategoričke promenljive Y1...Y18

# Zatvoreno polje : 0
# Broj 1 : 10
# Broj 2 : 20
# Broj 3 : 30
# Broj 4 : 40
# Broj 5 : 50
# Broj 6 : 60
# Broj 7 : 70
# Broj 8 : 80
# Mina : -100
# Prazno polje : 100


#plot(slike[[1]])
Y1 <-c(0,0,0,0,0,0,0,0,0,
       0,0,0,20,10,10,-100,10,0,
       0,0,30,-100,10,10,10,10,0,
       20,30,-100,20,10,100,10,10,0,
       -100,20,10,10,100,100,10,-100,0,
       10,10,100,100,10,10,20,10,0,
       100,10,10,10,10,-100,20,10,0,
       100,10,-100,0,0,0,0,0,0,
       100,10,0,0,0,0,0,0,0)

#plot(slike[[2]])
```

```
Y2 <-c(0,0,0,10,100,10,10,10,100,
       0,20,0,20,10,10,-100,10,100,
       0,0,30,-100,10,10,10,10,100,
       20,30,-100,20,10,100,10,10,10,
       -100,20,10,10,100,100,10,-100,10,
       10,10,100,100,10,10,20,10,10,
       100,10,10,10,10,-100,20,10,100,
       100,10,-100,10,10,20,0,10,100,
       100,10,10,10,100,10,0,10,100)

#plot(slike[[3]])
Y3 <-c(100,100,100,100,100,100,10,0,0,
       10,10,100,100,100,10,20,0,0,
       0,10,100,100,100,20,0,0,0,
       0,10,100,100,100,20,0,30,10,
       0,10,10,10,100,10,10,10,100,
       0,0,0,10,100,100,100,100,100,
       0,0,0,10,100,10,10,10,100,
       0,0,0,10,10,10,0,20,10,
       0,0,0,0,0,0,0,0,0)



#plot(slike[[4]])
Y4 <-c(100,100,100,100,100,100,100,10,0,
       10,10,100,100,100,100,100,10,0,
       0,10,100,100,10,20,30,30,0,
       20,20,100,100,10,-100,-100,-100,10,
       0,10,100,100,10,20,30,20,10,
       0,10,10,10,10,100,100,100,100,
       10,10,10,0,20,10,10,100,100,
       0,0,20,10,20,0,10,100,100,
       0,0,10,100,10,0,10,100,100)

#plot(slike[[5]])
Y5 <-c(100,100,100,100,100,100,100,100,100,
       100,100,100,100,100,10,10,10,100,
       100,100,100,100,100,20,0,30,10,
       10,10,100,100,100,20,0,0,0,
       0,10,100,10,20,30,30,30,0,
       10,10,100,10,0,0,10,10,0,
       100,10,10,20,20,20,10,10,10,
       10,20,0,10,100,100,100,10,0,
       0,20,10,10,100,100,100,10,0)

#plot(slike[[6]])
Y6 <-c(100,100,100,100,100,100,10,20,0,
       20,20,20,10,10,100,10,0,20,
       -100,-100,0,0,30,10,10,10,10,
       20,20,30,0,0,10,100,100,100,
       100,100,10,20,20,10,100,100,100,
       100,10,10,10,100,100,100,100,100,
       100,10,0,10,100,100,100,100,100,
       100,20,20,30,10,10,100,100,100,
```

```
       100,10,0,20,0,10,100,100,100)

#plot(slike[[7]])
Y7 <- c(100,100,100,100,10,0,20,10,100,
      100,100,100,100,10,30,0,20,100,
      100,100,100,100,100,20,0,20,100,
      100,100,100,100,100,10,20,20,10,
      10,10,10,10,10,100,10,0,10,
      0,0,0,0,10,100,10,10,10,
      0,0,0,0,20,10,100,100,100,
      0,0,0,0,0,10,10,10,10,
      0,0,0,0,0,0,0,0,0)

#plot(slike[[8]])
Y8 <-c(100,10,10,10,100,100,100,100,100,
       100,10,0,10,100,100,10,20,20,
       10,20,10,10,10,20,30,0,0,
       0,10,100,100,10,0,0,40,30,
       10,10,10,10,20,20,20,20,0,
       100,100,10,0,20,10,100,10,10,
       100,100,10,20,0,10,100,100,100,
       100,100,100,10,10,10,10,10,10,
       100,100,100,100,100,100,10,0,0)

#plot(slike[[9]])
Y9 <-c(0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,
       0,0,0,0,40,30,40,0,0,
       0,0,0,0,20,100,10,20,0,
       0,0,0,0,20,100,100,10,0,
       0,0,0,0,20,20,20,20,0,
       0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0)

#plot(slike[[10]])
Y10 <- c(0,0,0,0,0,0,0,0,0,
       0,30,10,30,0,0,0,0,0,
       0,20,100,10,20,20,0,0,0,
       10,10,100,100,100,10,0,0,0,
       100,100,100,10,20,40,0,0,0,
       10,20,10,20,0,0,0,0,0,
       0,0,0,0,0,0,0,0,0,
       0,0,0,0,0,0,0,40,0,
       0,0,0,0,0,0,0,0,0)
#plot(slike[[11]])
Y11 <- c(0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,30,0,
         0,0,-100,-100,-100,-100,30,0,0,
         0,0,40,40,30,30,30,0,0,
         0,0,-100,20,100,10,-100,30,-100,
         0,0,-100,20,100,10,10,20,10,
         0,0,40,20,100,100,100,10,10,
         0,-100,-100,20,10,10,100,10,-100,
```

```
         0,-100,30,20,-100,10,100,10,10)

#plot(slike[[12]])

Y12 <- c(0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,40,10,30,0,0,0,
         0,0,0,30,100,30,0,0,0,
         0,0,0,30,100,30,0,0,0,
         0,0,0,30,20,40,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0)

#plot(slike[[13]])

Y13 <- c(0,0,0,0,0,0,0,0,0,
         0,0,0,40,20,40,-100,0,0,
         0,0,0,20,100,30,-100,0,0,
         0,0,0,40,20,30,-100,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0)

#plot(slike[[14]])

Y14 <- c(0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,30,20,30,0,0,0,
         0,0,0,20,100,20,0,0,0,
         0,0,0,40,30,40,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0)

#plot(slike[[15]])
Y15 <- c(0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,40,30,30,0,0,
         0,0,0,0,10,100,20,0,0,
         0,0,0,0,40,30,40,0,0,
         0,0,0,0,0,0,0,0,0)

#plot(slike[[16]])

Y16 <- c(0,0,20,10,100,10,-100,0,0,
         0,0,-100,20,100,20,40,0,0,
         0,0,-100,30,100,10,-100,0,0,
```

```
         0,0,-100,30,20,30,40,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0,0,
         0,40,0,0,0,0,0,0,0)

#plot(slike[[17]])

Y17 <- c(100,100,100,100,10,0,10,100,100,
         100,100,100,100,10,10,20,10,10,
         100,100,100,100,100,100,10,0,10,
         10,10,10,100,100,100,10,10,10,
         20,0,10,100,100,100,10,10,10,
         0,40,30,10,100,100,10,0,0,
         20,0,0,30,30,20,20,10,10,
         10,20,30,0,0,0,10,100,100,
         100,100,10,20,30,20,10,0,0)
#plot(slike[[18]])
Y18 <- c(100,100,100,100,100,100,10,10,10,
        10,10,100,100,100,10,20,-100,20,
        -100,10,100,100,100,20,-100,40,-100,
        10,10,100,100,100,20,-100,30,10,
        100,10,10,10,100,10,10,10,100,
        100,10,-100,10,100,100,100,100,100,
        100,10,10,10,100,10,10,10,100,
        10,10,20,10,10,10,-100,20,10,
        10,-100,20,-100,10,10,10,20,-100)


#spajam ih sve u jedan vektor

Y=c(Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8,Y9,Y10,Y11,Y12,Y13,Y14,Y15,Y16,Y17,Y18)
```

Sledeći korak je pravljenje odgovarajućih prediktora.

Funkcija koja pravi prediktor koji razlikuje polja na osnovu vrednosti plave boje:

```
predictor<-function(im,vrste,kolone){

  X=c()
  l=1

   for(j in 1:(length(vrste)/2)){

     for(i in 1:(length(kolone)/2)){
      s=array(im[vrste[2*j-1]:vrste[2*j],kolone[2*i-1]:kolone[2*i],1,1:3],
              c(vrste[2*j]-vrste[2*j-1]+1,kolone[2*i]-kolone[2*i-1]+1,1,3)
)
      nova=cimg(s)
      red=as.vector(nova[,,1,1])
      green=as.vector(nova[,,1,2])
      blue=as.vector(nova[,,1,3])
      brojac=mean(blue/(red+green+blue), na.rm=TRUE)
      X[l]=brojac
```

```
      l=l+1
    }}
  return(X)

}
```

Funkcija koja pravi prediktor koji ih razlikuje na osnovu vrednosti disperzije polja u crno-belom spektru:

```
predictor1 <- function(slika,vrste,kolone){

  X <- c()
  l <- 1
  # r širina odsecanja
  r <- ceiling((vrste[2]-vrste[1])/16)
  for(j in 1:9){
    for(i in 1:9){
      s <-array(slika[(vrste[2*j-1]+r):(vrste[2*j]-r), (kolone[2*i-1]+r):(
kolone[2*i]-r), 1, 1:3],
                c(vrste[2*j]-vrste[2*j-1]-2*r+1,kolone[2*i]-kolone[2*i-1]-
2*r+1, 1, 3))
      X[l] <- var(as.vector(grayscale(cimg(s))[,,,1]))
      l <-l+1
    }
  }
  return(X)
}
```

Funkcija koja pravi prediktor koji razlikuje polja na osnovu najmanje vrednosti plave boje, slicno kao prvi prediktor

```
predictor2 <- function(slika, vrste, kolone)
{
  prosek_zbira <- c()

  l <- 1
  for(j in 1:9)
  {
    for(i in 1:9)
    {
      #izdvojimo polje
      s <- array(slika[vrste[2*j-1]:vrste[2*j], kolone[2*i-1]:kolone[2*i],
1, 1:3],
                 c(vrste[2*j]-vrste[2*j-1]+1,kolone[2*i]-kolone[2*i-1]+1,
1, 3))
      nova <- cimg(s)
      crvena <- as.vector(nova[ , , 1, 1])
      zelena <- as.vector(nova[ , , 1, 2])
      plava <- as.vector(nova[ , , 1, 3])

      brojac_zbira <- mean((crvena+zelena)/(crvena+zelena+plava),na.rm = T
RUE)
      prosek_zbira[l] <-brojac_zbira
```

```
      l=l+1
    }
  }

  return(prosek_zbira)
}
```

Sada pravimo prediktore za svaku sliku iz skupa za obučavanje:

```
X <-c()
X1 <-c()
X2 <-c()

for(i in 1: 18){

  slika=rm.alpha(slike[[i]]) #hoćemo da slike budu u RGB spektru
  slika=resize(slika,751,758,1,3)#podešavamo da sve slike budu istih dimen
zija
  q11 <-slika [ , , 1, 1]
  q12 <-slika [ , , 1, 2]
  q13 <-slika [ , , 1, 3]
  M <-pmax(q11, q12, q13) #pmax dodeljuje maksimalnu vrednost po elementim
a

  m_vrsta <-granice(M,0.2,0.3)
  m_kolona <-granice(M,0.2, 0.3,FALSE)

  v <-sredjivanje(m_vrsta)
  k <-sredjivanje(m_kolona)


  #za svaku sliku tražimo prediktor, i spajamo sve prediktore u jedan
  X <-c(X, predictor(slika, v, k))
  X1 <-c(X1,predictor1(slika,v,k))
  X2<-c(X2,predictor2(slika,v,k))
  }
```

Učitavamo slike iz kontrolnog skupa

```
setwd("C:\\Users\\Radulovic\\Desktop\\cetvrti_zadatak_kontrolni")
files <- list.files(path=getwd(),all.files=T, full.names=F, no.. = T)
slike_k <- lapply(files, load.image)
```

Pravimo kategoričku promenljivu za kontrolni skup

```
#plot(slike_k[[1]])
Yk1 <- c(10,10,20,-100,10,100,100,100,100,
            20,-100,30,10,10,100,100,100,100,
            20,-100,20,100,100,10,10,10,100,
            0,20,10,100,100,10,-100,20,10,
            -100,10,100,10,10,20,20,0,0,
            10,10,100,10,-100,10,10,0,0,
            100,100,100,10,10,10,10,20,0,
            100,100,100,100,10,10,20,0,0,
            100,100,100,100,10,0,20,0,0)
```

```
#plot(slike_k[[2]])
Yk2 <- c(0,0,10,0,0,-100,10,100,100,
            10,10,10,10,20,20,10,10,10,
            100,100,100,100,100,100,100,10,0,
            100,100,100,100,100,100,100,10,10,
            100,100,100,100,100,10,10,20,10,
            100,100,100,100,100,20,-100,30,-100,
            100,100,100,10,10,40,-100,40,10,
            100,100,100,20,-100,40,-100,20,100,
            100,100,100,20,-100,30,10,10,100)

#spajamo ih u jedan vektor
Yk <- c(Yk1, Yk2)
```

Zatim prediktore:

```
Xk=c()
X1k=c()
X2k<-c()

for(i in 1:length(slike_k)) {
  slika <- rm.alpha(slike_k[[i]])

  q11 <-slika [ , , 1, 1]
  q12 <-slika [ , , 1, 2]
  q13 <-slika [ , , 1, 3]
  M <- pmax(q11, q12, q13)

  m_vrsta2 <- granice(M,0.2, 0.3)
  m_kolona2<-granice(M,0.2, 0.3,FALSE)

  v2 <-sredjivanje(m_vrsta2)
  k2 <-sredjivanje(m_kolona2)

  Xk <- c(Xk, predictor(slika, v2, k2)) #dodajemo vrednost prediktora za o
vu sliku u vektore
  X1k <-c(X1k, predictor1(slika,v2,k2))
  X2k <-c(X2k,predictor2(slika,v2,k2))
  }
```

Prelazimo na modele.

QDA modeli sa različitim kombinacijama prediktora:

prediktori X1 i X2

```
model.qda<-qda(Y~X1+X2)

summary(model.qda)

##         Length Class  Mode
## prior    7     -none- numeric
## counts   7     -none- numeric
## means   14     -none- numeric
## scaling 28     -none- numeric
```

```
## ldet     7     -none- numeric
## lev      7     -none- character
## N        1     -none- numeric
## call     2     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
model.qda.pred<-predict(model.qda,newdata = data.frame(X1=X1k,X2=X2k))
table(model.qda.pred$class, Yk)
```

```
##       Yk
##        -100  0 10 20 30 40 100
##   -100   13  0  0  0  0  0   0
##   0       0 16  0  0  0  0   0
##   10      0  0 46  0  0  0   0
##   20      0  0  0 14  0  0   0
##   30      0  0  0  4  3  0   0
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0  63
```

```
mean(model.qda.pred$class == Yk)
```

```
## [1] 0.9753086
```

```
#pogađa oko 98%
```

prediktori X i X1

```
model.qda2<-qda(Y~X+X1)

model.qda2.pred<-predict(model.qda2,newdata = data.frame(X=Xk,X1=X1k))
table(model.qda2.pred$class, Yk)
```

```
##       Yk
##        -100  0 10 20 30 40 100
##   -100   13  0  0  0  0  0   0
##   0       0 16  0  0  0  0   0
##   10      0  0 46  0  0  0   0
##   20      0  0  0 14  0  0   0
##   30      0  0  0  4  3  0   0
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0  63
```

```
mean(model.qda2.pred$class == Yk)
```

```
## [1] 0.9753086
```

```
#pogađa oko 98%
```

Prediktor X (najveća vrednost plave boje)

```
model.qda.X<-qda(Y~X)

model.qdaX.pred<-predict(model.qda.X,newdata = data.frame(X=Xk))
table(model.qdaX.pred$class, Yk)
```

```
##        Yk
##         -100  0 10 20 30 40 100
##    -100    7  1  0  0  0  1   0
##    0       5 15  0  0  0  0   0
##    10      0  0 46  0  0  0   0
##    20      0  0  0 16  0  0   0
##    30      0  0  0  2  3  0   0
##    40      1  0  0  0  0  2   0
##    100     0  0  0  0  0  0  63
```

```
mean(model.qdaX.pred$class == Yk)
```

```
## [1] 0.9382716
```

```
#pogađa oko 94%
```

Prediktor X2 (vrednost zbira zelene i crvene)

```
model.qda.X2<-qda(Y~X2)

model.qdaX2.pred<-predict(model.qda.X2,newdata = data.frame(X2=X2k))
table(model.qdaX2.pred$class, Yk)
```

```
##        Yk
##         -100  0 10 20 30 40 100
##    -100    7  1  0  0  0  1   0
##    0       5 15  0  0  0  0   0
##    10      0  0 46  0  0  0   0
##    20      0  0  0 16  0  0   0
##    30      0  0  0  2  3  0   0
##    40      1  0  0  0  0  2   0
##    100     0  0  0  0  0  0  63
```

```
mean(model.qdaX2.pred$class == Yk)
```

```
## [1] 0.9382716
```

```
#pogađa oko 94%

#X i X2k imaju istu verovatnoću pogađanja, sto je logično
#gledajući kako su napravljeni
```

Prediktor X1 (disperzija polja u crno belom spektru)

```
model.qda.X1<-qda(Y~X1)

model.qdaX1.pred<-predict(model.qda.X1,newdata = data.frame(X1=X1k))
table(model.qdaX1.pred$class, Yk)
```

```
##        Yk
##         -100  0 10 20 30 40 100
##    -100   11  0  0  0  0  0   0
##    0       0 16  0  0  0  0  16
##    10      2  0 43  3  0  0   0
##    20      0  0  3 11  2  0   0
##    30      0  0  0  4  1  0   0
```

```
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0  47
```

```
mean(model.qdaX1.pred$class == Yk)
```

```
## [1] 0.8148148
```

```
#pogađa oko 81%
```

Posmatramo sada LDA model:

```
model.lda<-lda(Y~X1+X2)
summary(model.lda)
```

```
##         Length Class  Mode
## prior    7     -none- numeric
## counts   7     -none- numeric
## means   14     -none- numeric
## scaling  4     -none- numeric
## lev      7     -none- character
## svd      2     -none- numeric
## N        1     -none- numeric
## call     2     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
model.lda.pred=predict(model.lda,newdata = data.frame(X1=X1k,X2=X2k))
table(model.lda.pred$class,Yk)
```

```
##       Yk
##        -100  0 10 20 30 40 100
##   -100   13  0  0  0  0  0   0
##   0       0 16  0  0  0  0   0
##   10      0  0 46  0  0  0   0
##   20      0  0  0 14  2  0   0
##   30      0  0  0  4  1  0   0
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0  63
```

```
mean(model.lda.pred$class==Yk)
```

```
## [1] 0.962963
```

```
#96 posto pogađa
```

```
model.lda2<-lda(Y~X+X1)
summary(model.lda2)
```

```
##         Length Class  Mode
## prior    7     -none- numeric
## counts   7     -none- numeric
## means   14     -none- numeric
## scaling  4     -none- numeric
## lev      7     -none- character
## svd      2     -none- numeric
## N        1     -none- numeric
```

```
## call     2     -none- call
## terms    3     terms  call
## xlevels  0     -none- list

model.lda.pred2=predict(model.lda2,newdata = data.frame(X=Xk,X1=X1k))
table(model.lda.pred2$class,Yk)

##       Yk
##        -100  0 10 20 30 40 100
##   -100   13  0  0  0  0  0   0
##   0       0 16  0  0  0  0   0
##   10      0  0 46  0  0  0   0
##   20      0  0  0 14  2  0   0
##   30      0  0  0  4  1  0   0
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0  63

mean(model.lda.pred2$class==Yk)

## [1] 0.962963

#96%

model.ldaX2<-lda(Y~X2)
summary(model.ldaX2)

##         Length Class  Mode
## prior   7      -none- numeric
## counts  7      -none- numeric
## means   7      -none- numeric
## scaling 1      -none- numeric
## lev     7      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    2      -none- call
## terms   3      terms  call
## xlevels 0      -none- list

model.lda.predX2=predict(model.ldaX2,newdata = data.frame(X2=X2k))
table(model.lda.predX2$class,Yk)

##       Yk
##        -100  0 10 20 30 40 100
##   -100    7  2  0  0  0  3   0
##   0       2 14  0  0  0  0   0
##   10      4  0 46  0  0  0   0
##   20      0  0  0 18  3  0   0
##   30      0  0  0  0  0  0   0
##   40      0  0  0  0  0  0   0
##   100     0  0  0  0  0  0  63

mean(model.lda.predX2$class==Yk)

## [1] 0.9135802
```

```r
#91 posto pogadja

model.ldaX<-lda(Y~X)
summary(model.ldaX)

##         Length Class  Mode
## prior   7      -none- numeric
## counts  7      -none- numeric
## means   7      -none- numeric
## scaling 1      -none- numeric
## lev     7      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    2      -none- call
## terms   3      terms  call
## xlevels 0      -none- list

model.lda.predX=predict(model.ldaX,newdata = data.frame(X=Xk))
table(model.lda.predX$class,Yk)

##       Yk
##        -100  0 10 20 30 40 100
##   -100    7  2  0  0  0  3   0
##   0       2 14  0  0  0  0   0
##   10      4  0 46  0  0  0   0
##   20      0  0  0 18  3  0   0
##   30      0  0  0  0  0  0   0
##   40      0  0  0  0  0  0   0
##   100     0  0  0  0  0  0  63

mean(model.lda.predX$class==Yk)

## [1] 0.9135802

#91#

model.ldaX1<-lda(Y~X1)
summary(model.ldaX1)

##         Length Class  Mode
## prior   7      -none- numeric
## counts  7      -none- numeric
## means   7      -none- numeric
## scaling 1      -none- numeric
## lev     7      -none- character
## svd     1      -none- numeric
## N       1      -none- numeric
## call    2      -none- call
## terms   3      terms  call
## xlevels 0      -none- list

model.lda.predX1=predict(model.ldaX1,newdata = data.frame(X1=X1k))
table(model.lda.predX1$class,Yk)

##       Yk
##        -100  0 10 20 30 40 100
```

```
##   -100   13  0  2  0  0  0   0
##   0       0 16  0  0  0  0  58
##   10      0  0 41  3  0  0   0
##   20      0  0  3 11  2  0   0
##   30      0  0  0  4  1  0   0
##   40      0  0  0  0  0  3   0
##   100     0  0  0  0  0  0   5
```

```
mean(model.lda.predX1$class==Yk)
```

```
## [1] 0.5555556
```

```
#pogadja 55%
```

I na kraju posmatramo multinomni model:

```
model.multinomX<-multinom(Y~X)
```

```
## # weights:  21 (12 variable)
## initial  value 2837.136997
## iter  10 value 1212.811251
## iter  20 value 398.197805
## iter  30 value 372.942136
## iter  40 value 327.240897
## iter  50 value 309.863312
## iter  60 value 293.478941
## iter  70 value 284.853493
## iter  80 value 268.929222
## iter  90 value 260.999889
## iter 100 value 258.025257
## final  value 258.025257
## stopped after 100 iterations
```

```
summary(model.multinomX)
```

```
## Call:
## multinom(formula = Y ~ X)
##
## Coefficients:
##     (Intercept)          X
## 0     -63.58625  136.70697
## 10     62.70526 -148.41687
## 20    302.72048 -840.73929
## 30    241.15717 -642.68192
## 40    -20.76710   43.15049
## 100   183.31385 -467.82365
##
## Std. Errors:
##     (Intercept)        X
## 0      8.412845 17.83197
## 10     9.107537 22.40748
## 20    23.040662 67.37260
## 30    17.799062 47.82039
## 40     7.455118 15.96262
## 100   15.652051 40.68650
```

```
##
## Residual Deviance: 516.0505
## AIC: 540.0505
```

```
model.mulpredX <- predict(model.multinomX, newdata = data.frame(X=Xk))
table(model.mulpredX, Yk)
```

```
##                Yk
## model.mulpredX -100  0 10 20 30 40 100
##           -100    7  0  0  0  0  1   0
##           0       6 16  0  0  0  2   0
##           10      0  0 46  0  0  0   0
##           20      0  0  0 17  0  0   0
##           30      0  0  0  1  3  0   0
##           40      0  0  0  0  0  0   0
##           100     0  0  0  0  0  0  63
```

```
mean(model.mulpredX == Yk)
```

```
## [1] 0.9382716
```

```
#pogađa oko 94%
```

```
model.multinom<-multinom(Y~X+X1)
```

```
## # weights:  28 (18 variable)
## initial  value 2837.136997
## iter  10 value 1023.730406
## iter  20 value 204.455665
## iter  30 value 171.194879
## iter  40 value 144.718036
## iter  50 value 130.426794
## iter  60 value 59.738087
## iter  70 value 58.916863
## iter  80 value 50.190713
## iter  90 value 49.478012
## iter 100 value 47.877690
## final  value 47.877690
## stopped after 100 iterations
```

```
summary(model.multinom)
```

```
## Call:
## multinom(formula = Y ~ X + X1)
##
## Coefficients:
##     (Intercept)          X        X1
## 0      1.132952   24.54235 -690.9694
## 10    54.139594 -151.43296  346.3532
## 20   256.670592 -703.48628  -63.1605
## 30   103.291669 -315.92643  567.6437
## 40   -79.306135  128.23475  451.6549
## 100  102.296679 -239.61565 -610.5293
##
## Std. Errors:
```

```
##     (Intercept)         X       X1
## 0      25.55224  53.62374 258.6753
## 10     27.19281  57.31390 201.7306
## 20     86.47420 235.30388 347.1289
## 30     79.41828 213.53038 347.6261
## 40    335.78851 695.31640 306.7731
## 100    83.18613 224.63092 433.1825
##
## Residual Deviance: 95.75538
## AIC: 131.7554
```

```
model.mulpred <- predict(model.multinom, newdata = data.frame(X=Xk,X1=X1k)
)
table(model.mulpred, Yk)
```

```
##               Yk
## model.mulpred -100  0 10 20 30 40 100
##          -100   13  0  0  0  0  0   0
##          0       0 16  0  0  0  0   0
##          10      0  0 46  0  0  0   0
##          20      0  0  0 15  1  0   0
##          30      0  0  0  3  2  0   0
##          40      0  0  0  0  0  3   0
##          100     0  0  0  0  0  0  63
```

```
mean(model.mulpred == Yk)
```

```
## [1] 0.9753086
```

```
#pogađa oko 98%
```

```
model.multinom2<-multinom(Y~X1+X2)
```

```
## # weights:  28 (18 variable)
## initial  value 2837.136997
## iter  10 value 920.753096
## iter  20 value 225.155190
## iter  30 value 181.210514
## iter  40 value 155.832127
## iter  50 value 146.106566
## iter  60 value 63.699624
## iter  70 value 60.670785
## iter  80 value 57.989954
## iter  90 value 56.364348
## iter 100 value 53.034352
## final  value 53.034352
## stopped after 100 iterations
```

```
summary(model.multinom2)
```

```
## Call:
## multinom(formula = Y ~ X1 + X2)
##
## Coefficients:
##     (Intercept)         X1          X2
```

```
## 0      13.18232 -561.96046   -6.406761
## 10    -92.73616  309.69564  145.818827
## 20   -377.47533  -34.08898  589.258069
## 30   -149.56741  493.20081  218.655772
## 40     62.16449  392.74134 -149.568958
## 100  -120.88776 -611.19552  213.785831
##
## Std. Errors:
##     (Intercept)       X1        X2
## 0      15.52312 139.3967  32.27051
## 10     24.94465 169.3434  46.31346
## 20     59.77431 207.9148  94.65129
## 30     27.47215 192.2346  50.74883
## 40     72.11528 228.8038 148.20559
## 100   170.47656 353.3268 268.21048
##
## Residual Deviance: 106.0687
## AIC: 142.0687
```

```
model.mulpred2 <- predict(model.multinom2, newdata = data.frame(X1=X1k,X2=
X2k))
table(model.mulpred2, Yk)
```

```
##                Yk
## model.mulpred2 -100  0 10 20 30 40 100
##           -100   13  0  0  0  0  0   0
##           0       0 16  0  0  0  0   0
##           10      0  0 46  0  0  0   0
##           20      0  0  0 15  1  0   0
##           30      0  0  0  3  2  0   0
##           40      0  0  0  0  0  3   0
##           100     0  0  0  0  0  0  63
```

```
mean(model.mulpred2 == Yk)
```

```
## [1] 0.9753086
```

```
#pogađa oko 98%
```

```
model.multinomX1<-multinom(Y~X1)
```

```
## # weights:  21 (12 variable)
## initial  value 2837.136997
## iter  10 value 1746.964024
## iter  20 value 975.908842
## iter  30 value 875.538481
## iter  40 value 804.727431
## iter  50 value 632.122032
## iter  60 value 627.387598
## iter  70 value 611.687742
## iter  80 value 609.977803
## iter  90 value 609.167193
## iter 100 value 605.021532
```

```
## final  value 605.021532
## stopped after 100 iterations
```

```
summary(model.multinomX1)
```

```
## Call:
## multinom(formula = Y ~ X1)
##
## Coefficients:
##     (Intercept)         X1
## 0      28.12078 -1431.9338
## 10    -11.87213   450.2925
## 20    -30.96935   881.3550
## 30    -48.33567  1195.7761
## 40    -61.93627  1377.4189
## 100    29.74173 -2077.6905
##
## Std. Errors:
##     (Intercept)       X1
## 0      2.269072 23.45072
## 10     1.600622 56.26550
## 20     2.489851 71.67087
## 30     3.897805 91.12182
## 40     4.804021 99.38206
## 100    2.237871 23.46174
##
## Residual Deviance: 1210.043
## AIC: 1234.043
```

```
model.mulpredX1 <- predict(model.multinomX1, newdata = data.frame(X1=X1k))
table(model.mulpredX1, Yk)
```

```
##                Yk
## model.mulpredX1 -100  0 10 20 30 40 100
##            -100    7  0  0  0  0  0   0
##            0       0 16  0  0  0  0  16
##            10      6  0 45  4  0  0   0
##            20      0  0  1 11  2  0   0
##            30      0  0  0  3  1  0   0
##            40      0  0  0  0  0  3   0
##            100     0  0  0  0  0  0  47
```

```
mean(model.mulpredX1 == Yk)
```

```
## [1] 0.8024691
```

```
#pogađa oko 80%
```

```
model.multinomX2<-multinom(Y~X2)
```

```
## # weights:  21 (12 variable)
## initial  value 2837.136997
## iter  10 value 1201.846999
## iter  20 value 448.648496
```

```
## iter  30 value 392.609458
## iter  40 value 360.753826
## iter  50 value 328.816001
## iter  60 value 308.860281
## iter  70 value 293.694415
## iter  80 value 274.267315
## iter  90 value 266.544458
## iter 100 value 262.117420
## final  value 262.117420
## stopped after 100 iterations
```

```
summary(model.multinomX2)
```

```
## Call:
## multinom(formula = Y ~ X2)
##
## Coefficients:
##     (Intercept)         X2
## 0      74.66295 -139.59067
## 10    -76.61648  132.95766
## 20   -486.41138  761.87856
## 30   -372.19341  595.97510
## 40     23.48921  -45.16545
## 100  -264.11713  434.31960
##
## Std. Errors:
##     (Intercept)       X2
## 0      9.653064 18.25902
## 10    11.367661 19.13471
## 20    40.468905 61.01692
## 30    25.969567 41.35106
## 40     8.619798 16.15464
## 100   21.676763 35.20987
##
## Residual Deviance: 524.2348
## AIC: 548.2348
```

```
model.mulpredX2 <- predict(model.multinomX2, newdata = data.frame(X2=X2k))
table(model.mulpredX2, Yk)
```

```
##                Yk
## model.mulpredX2 -100  0 10 20 30 40 100
##            -100    7  0  0  0  0  1   0
##            0       6 16  0  0  0  2   0
##            10      0  0 46  0  0  0   0
##            20      0  0  0 17  0  0   0
##            30      0  0  0  1  3  0   0
##            40      0  0  0  0  0  0   0
##            100     0  0  0  0  0  0  63
```

```
mean(model.mulpredX2 == Yk)
```

```
## [1] 0.9382716
```

```
#pogađa oko 94%
```

Dobili smo sledeće rezultate predviđanja:

	X	X1	X2	X+X1	X1+X2
QDA	94%	81%	94%	98%	98%
LDA	91%	56%	91%	96%	96%
MULTINOM	94%	80%	94%	98%	98%

Zaključujemo da se najlošije pokazao model LDA, dok multinomni logistički i QDA imaju gotovo iste verovatnoće pogađanja.

Sva tri modela imaju dobar procenat pogađanja sa prediktorima X i X2, dok prediktor X1 daje slabije rezultate, naročito kod LDA modela. Ipak, kombinacije X1+X2 i X+X1 daju najbolje rezultate.

Zadatak 6.

Napisati sledeće funkcije:

a) prava matrica(matrica, dimenzija, broj_mina) - za prosleđenu već popunjenu do kraja (nema nijedne preostale neobeležene mine) matricu tabele Minolovca funkcija treba da vrati odgovor da li je ovakva matrica jedna moguća tabela minlovca.
b) generatortable(dimenzija, broj_mina) - ulazni argument dimenzije table (ne mora da bude kvadratna) i broj mina. Funkcija treba da vrati matricu - jednu gotovu i potpuno popunjenu tablu Minolovca.
v) sakrivanjepolja(matrica, broj_polja), koja za proizvoljnu matricu - tablu Minolovca, zatvori slučajno odabranih broj_poljapolja koja nisu mine.
g) MKsimulacija(matrica, ...)- za prosleđenu matricu tabele Minolovca funkcija treba da pomoću Monte Karlo simulacija odredi koje od preostalih polja ima najveću verovatnoću da sadrži minu i obratno, koje polje ima najmanju verovatnoću.

Rešenje:

a) pravimo funkciju koja proverava da li je prosledjena matrica prava tabla minolovca

```
#imamo prosleđen broj mina, proverićemo da li je on isti sa brojem mina na tabli
#ukoliko na tabli imamo neki broj, u okolnim poljima imamo tacno toliko mina
#ako imamo prazno polje, oko njega nema mina

prava_matrica<-function(matrica,dimenzija,broj_mina){

  P<-matrica

   #proširimo matricu da bi mogli od svakog polja da uzmemo sva okolna
  #i proverimo da li zadovoljavaju uslove
  #bitno je samo da uzmemo cifre koje se razlikuju od onih kojima kodiramo polja

  P<-cbind(rep(1,dimenzija),P,rep(1,dimenzija))
  P<-rbind(rep(1,(dimenzija+2)),P,rep(1,(dimenzija+2)))

  #proveravamo uslove samo ako je broj mina na tabli isti kao prosleđeni
   if((sum(matrica==-100))!=broj_mina){return(F)}

  else{

  #za svako polje pojedincačno, posmatramo okolna polja i proveravamo uslove
    for(i in 2:(dimenzija+1)){

      for(j in 2:(dimenzija+1)){

        okolna_polja=c(P[i-1,j-1],P[i-1,j],P[i-1,j+1],P[i,j-1],P[i,j+1],P[i+1,j-1],P[i+1,j],P[i+1,j+1])

        #prazno polje:
        if(P[i,j]==100){

          #broj mina u okolini je nula
          if(sum(okolna_polja==-100)!=0)

            return(FALSE)}

        #posmatramo polja sa brojem
        #broj na polju == broj mina u okolini
        for(k in 1:8){

           if(P[i,j]==10*k){

             if(sum(okolna_polja==-100)!=k)

               return(FALSE)

            }
```

```
        }

      }

    }
    return(T)}}
```

proveravamo na matrici M koja jeste prava tabla minolovca

```
M=matrix(c(100,100,100,100,100,10,-100,20,-100,
    10,10,100,100,100,10,10,20,10,
    -100,10,100,100,100,100,100,100,100,
    20,30,20,10,100,100,10,10,10,
    20,-100,-100,10,100,100,20,-100,20,
    -100,30,20,20,20,20,30,-100,20,
    10,10,100,10,-100,-100,20,10,10,
    100,100,100,10,20,20,10,100,100,
    100,100,100,100,100,100,100,100,100)  ,ncol=9,nrow = 9,
    byrow = T)

prava_matrica(M,9,10)  #vraća TRUE
```

```
## [1] TRUE
```

b) Pravimo funkciju koja za dat vektor dimenzija, i broj mina vraca jednu popunjenu tablu minolovca.

```
#postavljamo mine, potom popunjavamo matricu
#pošto matrica ne mora biti kvadratna, dimenzija je vektor koji sadrzi bro
j vrsta i broj kolona
generator_table<-function(dimenzija,broj_mina){

  v=dimenzija[1]
  k=dimenzija[2]
  tabla<-matrix(rep(0,v*k),nrow=v,ncol=k,byrow=T)


  #slučajno biramo pozicije gde su mine i postavljamo ih u matricu
  pozicije_mina=sample(1:(v*k),broj_mina,replace=F)

  for(i in 1:length(pozicije_mina)){

    tabla[pozicije_mina[i]]=-100

  }

  #sada popunjavamo ostatak matrice. Opet je proširujemo kao u delu pod a,
da bi svaki element imao 8 okolnih
  #polja, da olakšamo prolazak kroz matricu i proveravanje uslova
  P<-tabla
  P<-cbind(rep(1,v),P,rep(1,v))
  P<-rbind(rep(1,k+2),P,rep(1,k+2))
```

```
  #u for petlji prolazimo kroz sva polja table
  for(i in 2 : (v+1)){

    for(j in 2:(k+1)){

      okolna_polja=c(P[i-1,j-1],P[i-1,j],P[i-1,j+1],P[i,j-1],P[i,j+1],P[i+
1,j-1],P[i+1,j],P[i+1,j+1])

      #ako je u ovom polju mina, polje je već popunjeno, idemo na sledeće
polje
      if(P[i,j]==-100){

         next()
      }

      #proveravamo broj mina u okolini, ako je 0, P[i,j] je prazno polje
      if(sum(okolna_polja==-100)==0) {

         P[i,j]=100 }

      else{

        s=sum(okolna_polja==-100)
        P[i,j]=10*s

      }

    }

   }

  return(P[2:(v+1),2:(k+1)]) #skidamo vrste I kolone koje smo dodali i vra
ćamo matricu
  }
```

v) Pravimo funkciju koja zatvara slučajno odabranih broj_polja koja nisu mine

```
sakrivanje_polja<-function(matrica,broj_polja){


nova<-matrica

  #čuvamo sve pozicije da bi mogli da izdvojimo dozvoljene,i pozicije mina
  sve_pozicije=1:length(nova)
  pozicije_mina=which(nova==-100)
  dozvoljene_pozicije=sve_pozicije[-pozicije_mina]

  #slučajno biramo broj polja koja nisu mine koja zatvaramo
```

```
  zatvorena_polja=sample(dozvoljene_pozicije,broj_polja,replace=F)
  nova[zatvorena_polja]=0
  nova[pozicije_mina]=0
  return(nova)

}
```

g) Prvo pravimo pomoćnu funkciju koja tablu minilovca koja nije skroz popunjena, ali su otvorene sve mine, popunjava do kraja

```
popuni_kompletno=function(matrica){

  dimenzija=length(matrica[1,])
  P<-matrica

  #proširujemo matricu kao i ranije
  P<-cbind(rep(1,dimenzija),P,rep(1,dimenzija))
  P<-rbind(rep(1,(dimenzija+2)),P,rep(1,(dimenzija+2)))

  for(i in 2:(dimenzija+1)){

     for(j in 2:(dimenzija+1)){

      #popunjavamo samo zatvorena polja, da bismo mogli da proverimo da li
je matrica koju dobijemo prava
       if(P[i,j]==0){
        okolna_polja=c(P[i-1,j-1],P[i-1,j],P[i-1,j+1],P[i,j-1],P[i,j+1],P[
i+1,j-1],P[i+1,j],P[i+1,j+1])


        if(sum(okolna_polja==-100)==0){
          P[i,j]=100

        }
        else{
          P[i,j]=10*sum(okolna_polja==-100)

        }

       }
     }
  }
  return(P[2:(dimenzija+1),2:(dimenzija+1)])
}
```

Sada radimo Monte Karlo simulaciju:

```
#Neka je X broj vec otvorenih polja na kojima su mine. Ideja je da na sluč
ajan nacin, od preostalih polja izaberemo
#broj mina-X polja na koje smeštamo preostale mine. Zatim popunimo matricu
do kraja funkcijom popuni_kompletno, a
#onda funkcijom prava_matrica proveravamo da li je tabla koju smo mi dobil
i prava tabla minolovca( to je tačno samo ako smo mine
#rasporedili samo tamo gde smeju da budu). Ukoliko je to tačno, povećavamo
verovatnoću da se na poljima koja smo izabrali nalazi mina.
#uradimo simulaciju n puta, i na kraju vratimo pozicije sa najvećom,i najm
anjom verovatnoćom da je na njima mina.

montekarlo_simulacija<-function(matrica,broj_mina,...){


  M<- matrica
  dimenzija=length(M[1,])
  preostale_mine=broj_mina-sum(M==-100)
  pozicije=which(M==0)
  verovatnoce_pozicija=rep(0,length(pozicije))

 #n biramo tako da bude dovoljno veliko za bilo koji broj mina i pozicija.
 #Zato možemo uzeti za n k*10, gde je k broj načina na koji možemo raspore
diti preostale mine
 #na naše pozicije. To radimo jer imamo proizvoljne argumente, pa da budem
o sigurni da smo simulaciju
 #uradili dovoljno puta.
  n=(choose(length(pozicije),preostale_mine))*10

  for(i in 1:n){

    #slučajno raspoređujemo preostale mine na moguća polja
    s=sample(pozicije,preostale_mine,replace=F)
    M1=M
    M1[s]=-100
    popunjena=popuni_kompletno(M1)

    #proverićemo da li je matrica koju smo popunili prava
    if(prava_matrica(popunjena,dimenzija,broj_mina))

      for(j in s){
        verovatnoce_pozicija[which(pozicije==j)]=verovatnoce_pozicija[whic
h(pozicije==j)]+1

      }

  }

   max_pozicija=pozicije[which.max(verovatnoce_pozicija)]
   min_pozicija=pozicije[which.min(verovatnoce_pozicija)]
```

```
  return(c(max_pozicija,min_pozicija))}
```

Na kraju možemo proveriti simulaciju koisteći funkcije koje smo prethodno napisali.

```
 N<- sakrivanje_polja(M,5)
 N[3]=-100
 N[6]=-100
 N
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
##  [1,]  100  100  100    0    0   10    0   20    0
##  [2,]   10   10  100  100  100    0   10   20   10
##  [3,] -100   10  100  100  100  100    0  100  100
##  [4,]   20   30   20   10  100  100   10   10   10
##  [5,]   20    0    0   10  100  100   20    0   20
##  [6,] -100   30   20   20    0   20   30    0   20
##  [7,]   10   10  100   10    0    0   20   10   10
##  [8,]  100  100  100   10   20   20   10  100  100
##  [9,]  100  100  100  100  100  100  100  100  100
```

```
 c=montekarlo_simulacija(N,10)
 c
```

```
## [1] 14 28
```

```
 M1=matrix(c(100,0,0,0,
     0,0,0,0,
     0,0,0,0,
     0,0,0,1),nrow=4)
 M1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  100    0    0    0
## [2,]    0    0    0    0
## [3,]    0    0    0    0
## [4,]    0    0    0    1
```

```
 c=montekarlo_simulacija(M1,3)
 c
```

```
## [1] 11  2
```

Prvi element vektora koji vraća simulacija je pozicija na kojoj je najveća verovatnoća da bude mina, a drugi element pozicija za koju je ta verovatnoća najmanja.

Dakle, za matricu N, najveću verovatnoću da sadrži minu ima 14. pozicija, a najmanju 28.

Za M1 su te pozicije 11. i 2.