

Projekat iz predmeta prepoznavanje oblika

Zadatak 1

Projektovati inovativni sistem za prepoznavanje pokazanih znakova zasnovan na testiranju hipoteza.

- Detaljno opisati algoritam za obradu slike i odabir obeležja koji prethodi samoj klasifikaciji. Algoritam treba da bude što robustniji (na različite osvetljaje, položaje šaka, načine pokazivanja znakova itd).
- Izvršiti podelu na trening i test skup. Rezultate klasifikacije test skupa prikazati u obliku matrice konfuzije.
- Odabrati dva oblika i dva obeležja takva da su odabrani oblici što separabilniji u tom prostoru. Prikazati histogram obeležja za oba slova i prokomentarisati njihov oblik.
- Za slova i obeležja pod c) projektovati parametarski klasifikator po izboru i iscrtati klasifikacionu liniju

```
clear ;
close all;
clc;
imds_scissors = imageDatastore('C:/Users/Dragana/Downloads/P0 vezbe/data/scissors/*.png');
imds_paper = imageDatastore('C:/Users/Dragana/Downloads/P0 vezbe/data/paper/*.png');
imds_rock = imageDatastore('C:/Users/Dragana/Downloads/P0 vezbe/data/rock/*.png');
imgs_scissors = readall(imds_scissors);
imgs_paper = readall(imds_paper);
imgs_rock = readall(imds_rock);
```

```
img_paper = imgs_paper{1};
img_rock = imgs_rock{1};
img_scissors = imgs_scissors{1};
fig_id = 1;
figure(fig_id)
subplot(131);imshow(img_paper, [0,255]);title('Papir');
subplot(132);imshow(img_rock, [0,255]);title('Kamen');
subplot(133);imshow(img_scissors, [0,255]);title('Makaze');
sgtitle('Prikaz izgleda ulaznih slika u RGB')
```

Prikaz izgleda ulaznih slika u RGB

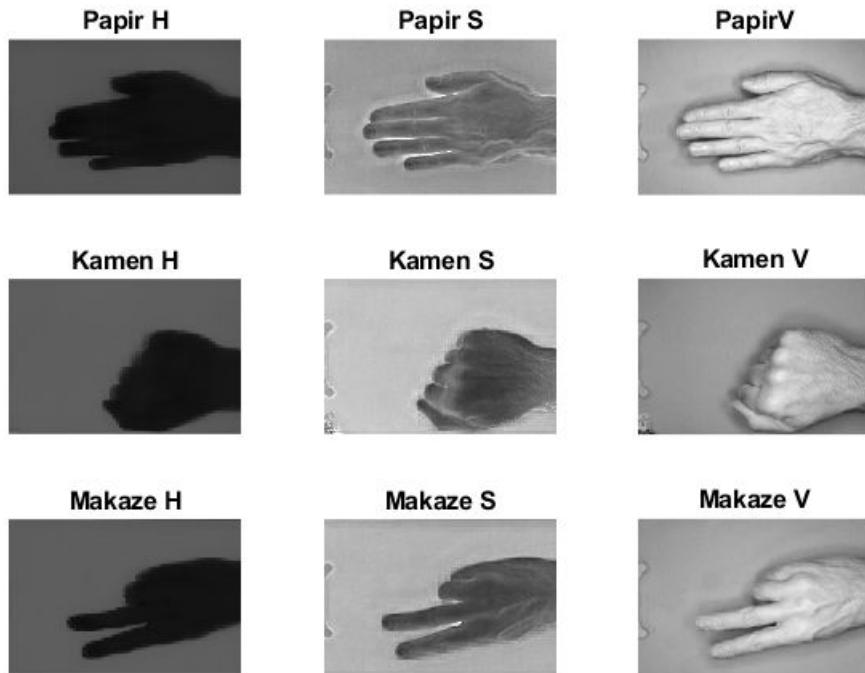


```
fig_id = fig_id+1;
```

S obzirom da je ruka coveka bez boje, ona ce se bojom uvek odvajati u odnosu na okolinu koje god boje da je sto ili odeca. Iz ovog razloga prelazimo u HSV sistem i ogranicavamo H komponentu kako bismo segmentisali deo gde je ruka. Pomocu ovako segmentisane slike mozemo izdvojiti i ivice ruke sto nam moze pomoci za izdvajanje bitnih obelezja. Ne zelimo da koristimo V komponentu jer je ona zapravo osvetljaj, pa pri razlicitim osvetljajima moze da se desi da nas algoritam radi losije.

```
img_paper_hsv = rgb2HSV(img_paper);
img_rock_hsv = rgb2HSV(img_rock);
img_scissors_hsv = rgb2HSV(img_scissors);
figure(fig_id)
subplot(331);imshow(img_paper_hsv(:,:,1));title('Papir H');
subplot(332);imshow(img_paper_hsv(:,:,2));title('Papir S');
subplot(333);imshow(img_paper_hsv(:,:,3));title('Papir V');
subplot(334);imshow(img_rock_hsv(:,:,1));title('Kamen H');
subplot(335);imshow(img_rock_hsv(:,:,2));title('Kamen S');
subplot(336);imshow(img_rock_hsv(:,:,3));title('Kamen V');
subplot(337);imshow(img_scissors_hsv(:,:,1));title('Makaze H');
subplot(338);imshow(img_scissors_hsv(:,:,2));title('Makaze S');
subplot(339);imshow(img_scissors_hsv(:,:,3));title('Makaze V');
sgtitle('Prikaz izgleda ulaznih slika u HSV')
```

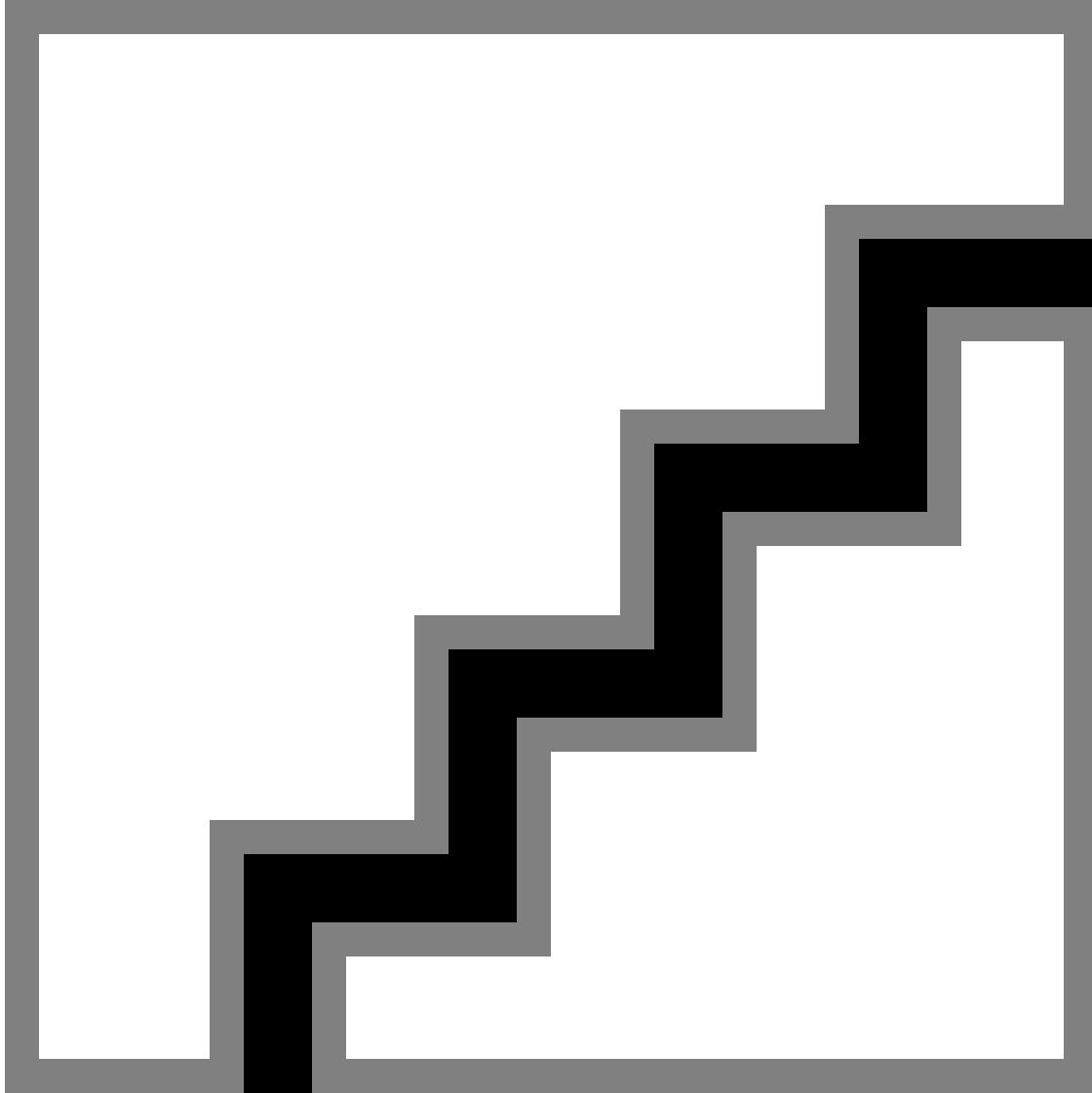
Prikaz izgleda ulaznih slika u HSV



```
fig_id = fig_id+1;
```

S obzirom da se boja koze nalazi oko 30 stepeni kada posmatramo H vrednosti, ogranicicemo H vrednost izmedju 20 i 40 stepeni tj. ako posmatramo H komponentu od 0 do 1 - izmedju 0.0556 i 0.1111. Ovo se moze videti i sa histograma H komponenti da se manji pik nalazi na sve tri otprilike u tom opsegu vrednosti.

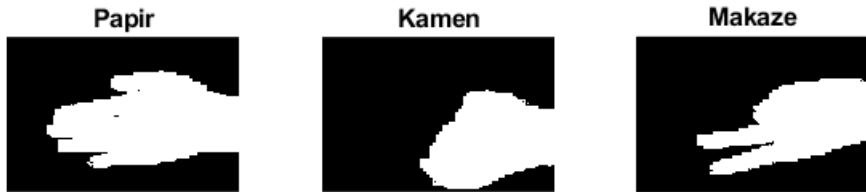
```
img_paper_h = img_paper_hsv(:,:,1);
img_rock_h = img_rock_hsv(:,:,1);
img_scissors_h = img_scissors_hsv(:,:,1);
figure(3)
subplot(131);histogram(img_paper_h); title('Papir');
subplot(132);histogram(img_rock_h); title('Kamen');
subplot(133);histogram(img_scissors_h); title('Makaze');
sgtitle('Prikaz histograma H komponenti ulaznih slika')
```



Sa histograma takodje mozemo primetiti da se broj piksela koji pripadaju ruci kod papira znatno izdvaja u odnosu na kamen i makaze. Iz tog razloga, to ce biti korisceno kao prvo obelezje pri klasifikaciji. Ovo obelezje nije skroz reprezentativno jer razlicite osobe imaju razlicitu velicinu ruke, ali dobro ce raditi u slucaju kada su ruke osoba za koje su slikane slike slicne velicine.

```
img_paper_seg = (img_paper_h>=0.04) & (img_paper_h<=0.12);
img_rock_seg = (img_rock_h>=0.04) & (img_rock_h<=0.12);
img_scissors_seg = (img_scissors_h>=0.04) & (img_scissors_h<=0.12);
figure(fig_id)
subplot(131);imshow(img_paper_seg,[0,1]); title('Papir');
subplot(132);imshow(img_rock_seg,[0,1]); title('Kamen');
subplot(133);imshow(img_scissors_seg,[0,1]); title('Makaze');
sgtitle('Prikaz izgleda segmentisanih ulaznih slika')
```

Prikaz izgleda segmentisanih ulaznih slika



```
fig_id = fig_id+1;
```

Primetimo da je na prvoj slici ostala crna tackica. Zbog ovakvih situacija, da ne bismo imali rupu u segmentu primenicemo dilataciju. Ova operacija dodeljuje svakom pikselu maksimum njegove okoline cime se popunjavaju sve rupe.

```
se = strel("rectangle",[3 3])
```

```
se =
strel is a rectangle shaped structuring element with properties:
```

```
    Neighborhood: [3x3 logical]
    Dimensionality: 2
```

```
img_paper_seg = imdilate(img_paper_seg,se);
img_rock_seg= imdilate(img_rock_seg,se);
img_scissors_seg = imdilate(img_scissors_seg,se);
figure(fig_id)
subplot(131);imshow(img_paper_seg,[0,1]); title('Papir');
subplot(132);imshow(img_rock_seg,[0,1]); title('Kamen');
subplot(133);imshow(img_scissors_seg,[0,1]); title('Makaze');
sgtitle('Prikaz izgleda segmentisanih ulaznih slika nakon dilatacije')
```

Prikaz izgleda segmentisanih ulaznih slika nakon dilatacije

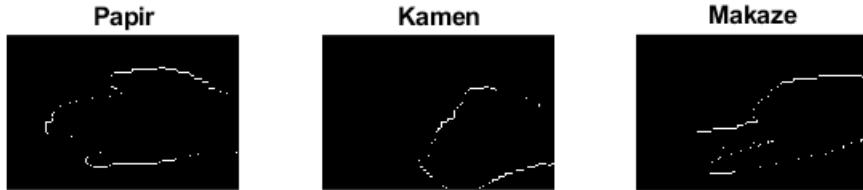


```
fig_id = fig_id+1;
```

Naredni korak je izdvajanje ivica.

```
BW_paper = edge(img_paper_seg);
BW_rock = edge(img_rock_seg);
BW_scissors = edge(img_scissors_seg);
figure(fig_id)
subplot(131);imshow(BW_paper,[0,1]); title('Papir');
subplot(132);imshow(BW_rock,[0,1]); title('Kamen');
subplot(133);imshow(BW_scissors,[0,1]); title('Makaze');
sgtitle('Prikaz izgleda ivica ruke ulaznih slika')
```

Prikaz izgleda ivica ruke ulaznih slika



```
fig_id = fig_id+1;
```

Sledeće najintuitivnije obelezje koje mozemo iz ovoga izvuci je broj ivičnih piksela, ipak i ovo obelezje ce zavisiti od ruke, pa da bismo uklonili zavisnost od velicine ruke mozemo taj broj podeliti sa površinom ruke da bismo dobili relativnu velicinu ivica. Ovo obelezje bi trebalo da odvoji makaze i papir od kamena.

Naredna obelezja su najveća rasirenost ruke koja bi trebala da odvoji papir od ostalih oblika. Poslednja dva obelezja su suma piksela u kolonama koje sadrže vise od dva piksela sto bi trebalo nekako da uhvati slike gde ima vise prstiju i razlika najveceg i najmanjeg indeksa kolone koja ima vise od dva piksela sto bi trebalo da predstavlja duzinu prstiju ako se vide.

```
% Function res = features(img, C)
res = zeros(C,1);
% preprocessing
img_hsv = rgb2hsv(img);
img_h = img_hsv(:,:,1);
img_t = (img_h>0.04) & (img_h<0.12);
se = strel("rectangle", [3 3]);
img_t = imdilate(img_t, se);
BW = edge(img_t);
edge_ids = find(BW==1);
rows = mod(edge_ids, 200);
cols = (edge_ids-rows)/200+1;
[cont_cols, unique_cols] = hist(cols, unique(cols));

% % area of hand
% res(1) = sum(sum(img_t));
% % sum of edges relative to size of hand
% res(2) = sum(sum(BW))/sum(sum(img_t));
% % max width of hand (paper should have bigger values for this feature)
```

```

% res(2) = sum(sum(BW))/sum(sum(img_t));
% & max width of hand (paper should have bigger values for this feature)
% res(3) = min((max(rows)-min(rows))+(max(cols)-min(cols)));
% & maximum distance from hand to end of matrix
% res(4) = max(max(200*max(rows),min(rows))+max(min(cols), 300 - max(cols)));

% area of hand
res(1) = sum(sum(img_t));
% sum of edges relative to size of hand
res(2) = sum(sum(BW))/sum(sum(img_t));
% max width of hand
res(3) = min((max(rows)-min(rows))+(max(cols)-min(cols)));
% sum of pixels in columns which consists of more than 2 pixels -> fingers
res(4) = sum(cnt_cols(cnt_cols>2));
% max distance between finger pixels
res(5) = max(unique_cols(cnt_cols>2))-min(unique_cols(cnt_cols>2));

```

```

C = 5;
P = zeros(C, length(imgs_paper));
for i = 1:length(imgs_paper)
    P(:,i) = features(imgs_paper{i},C);
end
S = zeros(C, length(imgs_scissors));
for i = 1:length(imgs_scissors)
    S(:,i) = features(imgs_scissors{i},C);
end
R = zeros(C, length(imgs_rock));
for i = 1:length(imgs_rock)
    R(:,i) = features(imgs_rock{i},C);
end

```

Odvajanje podataka na test i trening skup:

```

%shuffle
R = R(:, randperm(size(R, 2)));
% 30% test
R_test = R(:,1:200);
% 70% train
R_train = R(:,201:end);
%shuffle
P = P(:, randperm(size(P, 2)));
% 30% test
P_test = P(:,1:200);
% 70% train
P_train = P(:,201:end);
% shuffle
S = S(:, randperm(size(S, 2)));
% 30% test
S_test = S(:,1:200);
% 70% train
S_train = S(:,201:end);

```

Racunanje parametara i treniranje sa Bajesovim testom vise hipoteza se radi nad trening skupom. Bajesov test minimalne verovatnoce greske klasificuje podatke tako sto racuna aposteriornu fgv, i onda je klasa koja se predikuje ona cija je aposteriorna verovatnoca maksimalna.

```
Mp = mean(P_train,2);
Sp = cov(P_train');
Ms = mean(S_train,2);
Ss = cov(S_train');
Mr = mean(R_train,2);
Sr = cov(R_train');
```

```
Xs = [P_test, R_test, S_test];
Y_true = [ones(1,200),2*ones(1,200), 3*ones(1,200)];
Y_pred = zeros(size(Y_true));
constp = 1/(2*pi)/(det(Sp)^0.5);
consts = 1/(2*pi)/(det(Ss)^0.5);
constr = 1/(2*pi)/(det(Sr)^0.5);
Pp = 1/3%length(P)/(length(P)+length(R)+length(S));
```

Pp = 0.3333

```
Pr = 1/3%length(R)/(length(P)+length(R)+length(S));
```

Pr = 0.3333

```
Ps = 1/3%length(S)/(length(P)+length(R)+length(S));
```

Ps = 0.3333

```
for i = 1:length(Xs)
X = Xs(:,i);

f1 = constp*exp(-0.5*(X-Mp)']*inv(Sp)*(X-Mp));
f2 = constr*exp(-0.5*(X-Mr)']*inv(Sr)*(X-Mr));
f3 = consts*exp(-0.5*(X-Ms)']*inv(Ss)*(X-Ms));

q1 = Pp*f1/(Pp*f1+Pr*f2+Ps*f3);
q2 = Pr*f2/(Pp*f1+Pr*f2+Ps*f3);
q3 = Ps*f3/(Pp*f1+Pr*f2+Ps*f3);

[q,Y_pred(i)] = max([q1,q2,q3]);
```

end

```
ConMat= confusionmat(Y_true, Y_pred);
disp(['Confusion matrix on test set'])
```

Confusion matrix on test set

```
disp(ConMat)
```

```
138    48    14  
12     179   9  
10     1     189
```

```
disp('Accuracy')
```

```
Accuracy
```

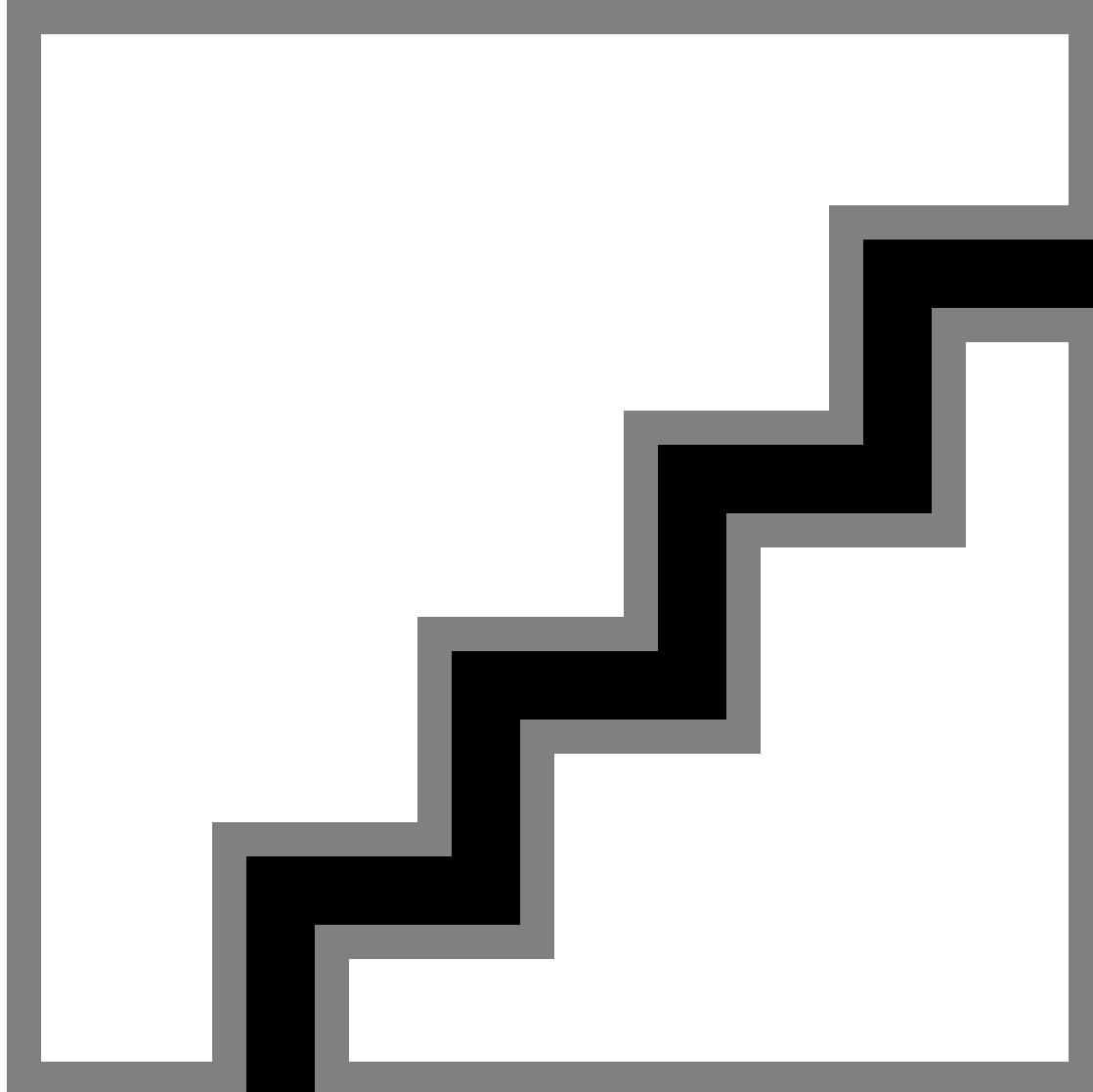
Vidimo da ovakav model daje zadovoljavajuće rezultate, ali da najviše gresi u slučaju papira što ima smisla jer je papir najteže odvojiti od ostalih oblika jer može da se javi sa rasirenim prstima kada lici na makaze i sa skupljenim prstima kada lici na kamen. Takođe, naš model najčešće mesa papir i kamen.

```
disp(sum(diag(ConMat))/(length(P_test)+length(R_test)+length(S_test)))
```

```
0.8433
```

Sa histograma možemo primetiti da prvo obeležje dobro odvaja papir od ostalih oblika, drugo dobro odvaja makaze od ostalih oblika, dok treće po malo odvaja sva tri, četvrto odvaja makaze od ostalih a peto kamen od ostalih.

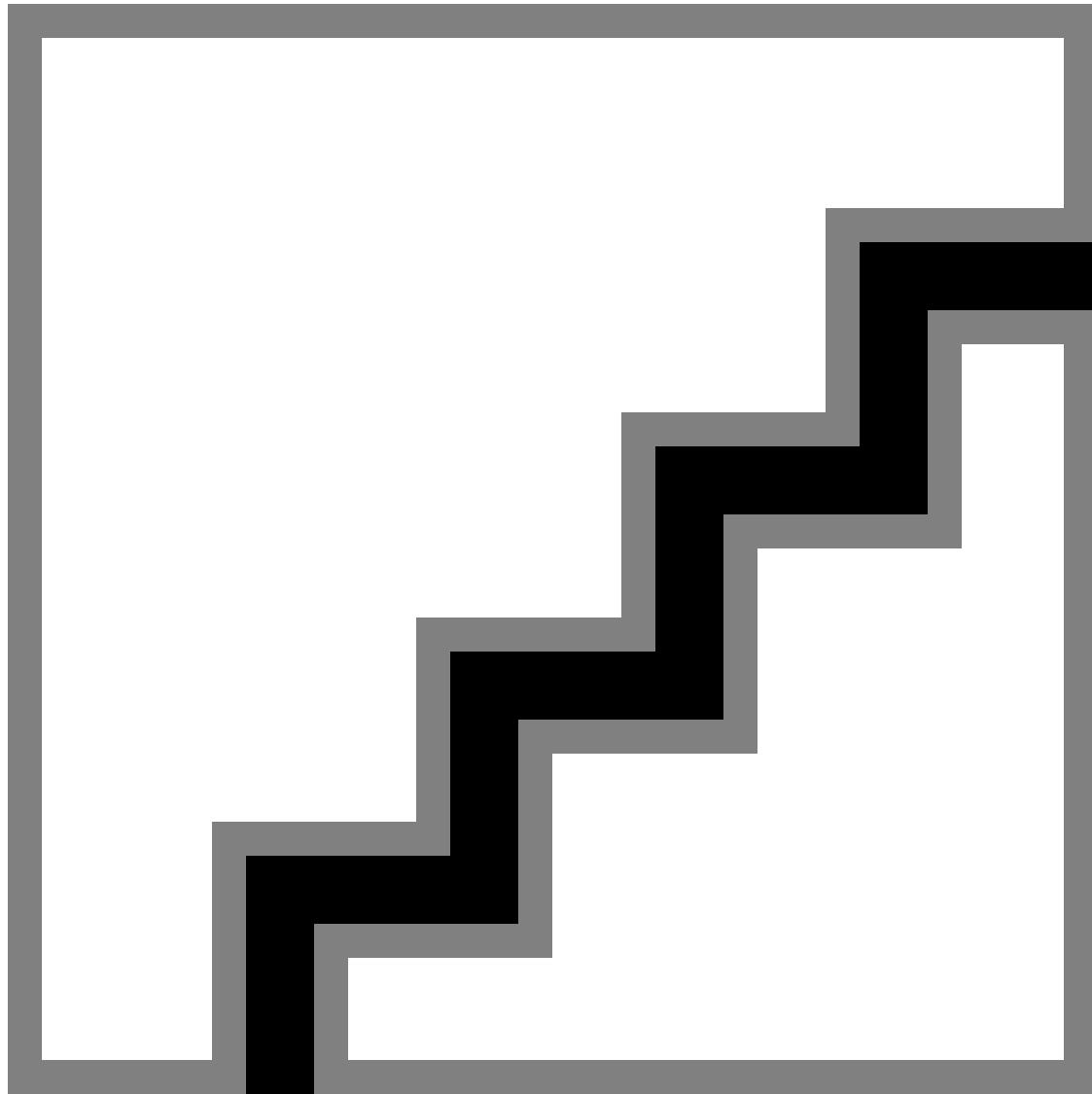
```
figure(fig_id)  
for i = 1:C  
    subplot(2,3,i);  
    histogram(R(i,:));  
    hold on;  
    histogram(S(i,:));  
    hold on;  
    histogram(P(i,:));  
    legend('R','S','P');  
    ttl = ['Obeležje ' int2str(i)];  
    xlabel('vrednost obeležja')  
    ylabel('broj odbiraka')  
    title(ttl);  
end  
sgtitle('Prikaz histograma svih obeležja')
```



```
fig_id = fig_id+1;
```

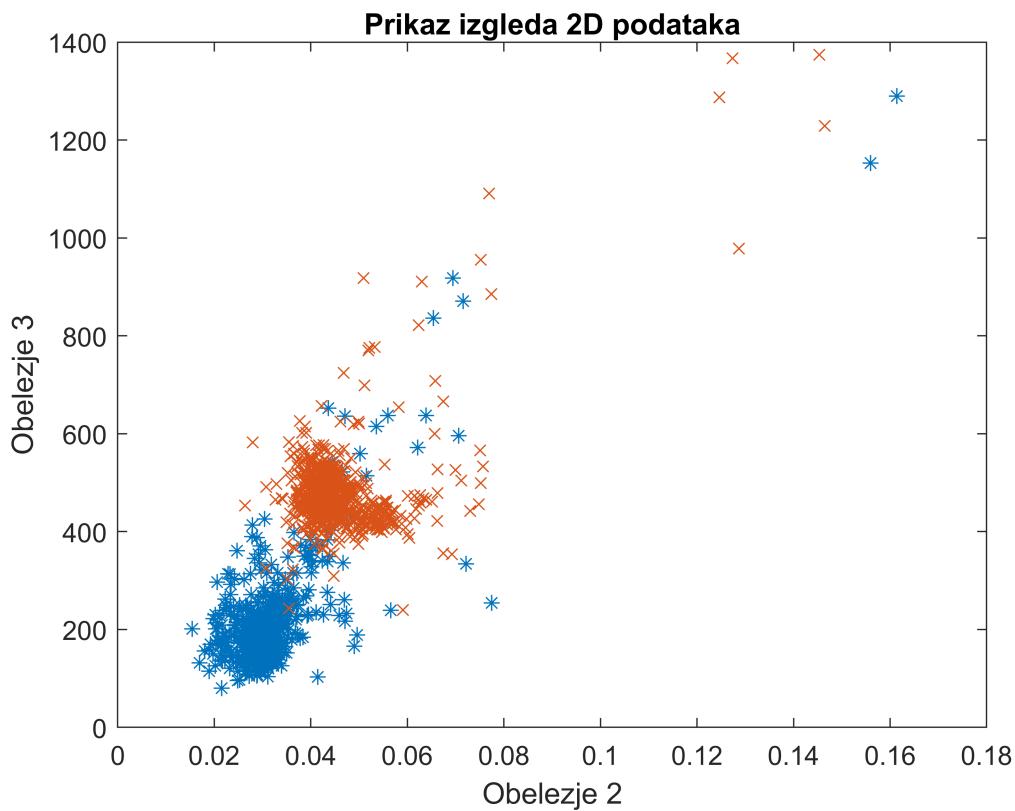
Kao dva dobro odvojena oblika biramo makaze i kamen, a kao dva obelezja koja ih dobro odvajaju biramo obelezja 2 i 4.

```
figure(fig_id)
subplot(121);histogram(S(4,:));hold on;histogram(R(4,:));
title('Obelezje 4');legend('S','R')
xlabel('vrednost obelezja')
ylabel('broj odbiraka')
subplot(122);histogram(S(2,:));hold on;histogram(R(2,:));
title('Obelezje 2'); legend('S','R')
sgtitle('Prikaz dva odabrana obelezja')
xlabel('vrednost obelezja')
ylabel('broj odbiraka')
```



```
fig_id = fig_id+1;
```

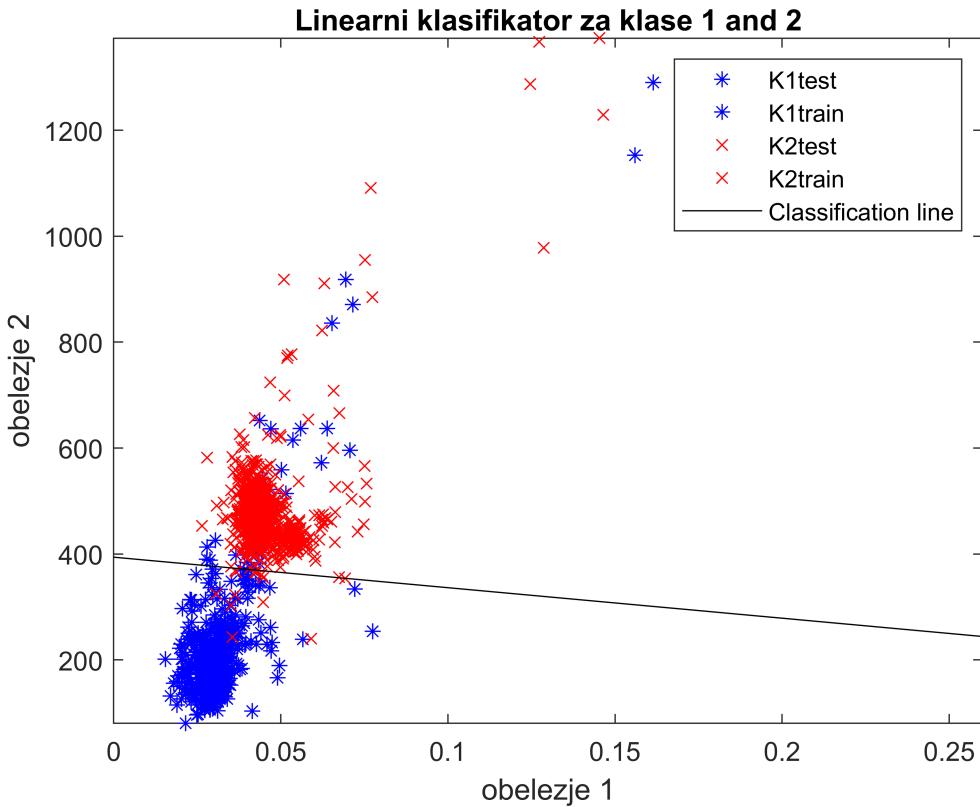
```
figure(fig_id)
plot(R(2,:), R(4,:), '*');
hold on;
plot(S(2,:), S(4,:), 'x');
title('Prikaz izgleda 2D podataka')
xlabel('Obelezje 2')
ylabel('Obelezje 3')
```



```
fig_id = fig_id+1;
```

Za parametarsku klasifikaciju koristimo linearni klasifikator o kome ce vise biti reci u 3. zadatku.

```
x1=0:0.01:0.26;
x2 = 200:15:600;
[V,v0] = LinearClassifier([R_test(2,:);R_test(4,:)], [R_train(2,:);R_train(4,:)], [S_test(2,:);S_train(4,:)]);
```



```

fig_id = fig_id+1;
h = (V'*[R_test(2,:);R_test(4,:)], [S_test(2,:);S_test(4,:)]]+v0);
y_pred = zeros(1,length(P_test));
y_pred(h<=0) = 1;
y_pred(h>0) = 2;

y_true = [ones(1,200),2*ones(1,200)];
C = confusionmat(y_true, y_pred);
disp(C)

```

| | |
|-----|-----|
| 190 | 10 |
| 3 | 197 |

Zadatak 2

Generisati po $n = 500$ odbiraka iz dveju dvodimenzionih bimodalnih klasa: $\Omega_1 \sim 11 \cdot (11, \Sigma_{11}) + 12 \cdot (12, \Sigma_{12})$, $\Omega_2 \sim 21 \cdot (21, \Sigma_{21}) + 22 \cdot (22, \Sigma_{22})$. Parametre klasa samostalno izabrati.

- Na dijagramu prikazati odbirke.
- Isrtati kako teorijski izgledaju funkcije gustine verovatnoće za raspodele klasa i uporediti ih sa histogramom generisanih odbiraka.
- Projektovati Bajesov klasifikator minimalne greške i na dijagramu, zajedno sa odbircima, skicirati klasifikacionu liniju. Uporediti grešku klasifikacije konkretnih odbiraka sa teorijskom greškom klasifikacije prve i druge vrste za datu postavku.
- Projektovati klasifikator minimalne cene tako da se više penalizuje pogrešna klasifikacija odbiraka iz prve klase. e) Ponoviti prethodnu tačku za Neuman-Pearson-ov klasifikator. Obrazložiti izbor $\lambda = 0$.

- Za klase oblika generisanih u prethodnim tačkama, projektovati Wald-ov sekvencijalni test pa skicirati zavisnost broja potrebnih odbiraka od usvojene verovatnoće grešaka prvog, odnosno drugog tipa.

Prikaz odbiraka

```

N = 500;
M11 = [0 0]';
S11 = [1 0.5; 0.5 1];
M12 = [0 8]';
S12 = [1 -0.2; -0.2 1];

M21 = [5 0]';
S21 = [1.5 -0.7; -0.7 1.5];
M22 = [5 6]';
S22 = [1 0.6; 0.6 1];

rng(20)

% first distribution
pom1 = rand(N, 1);
K11 = mvnrnd(M11,S11, N);
K12 = mvnrnd(M12,S12,N);

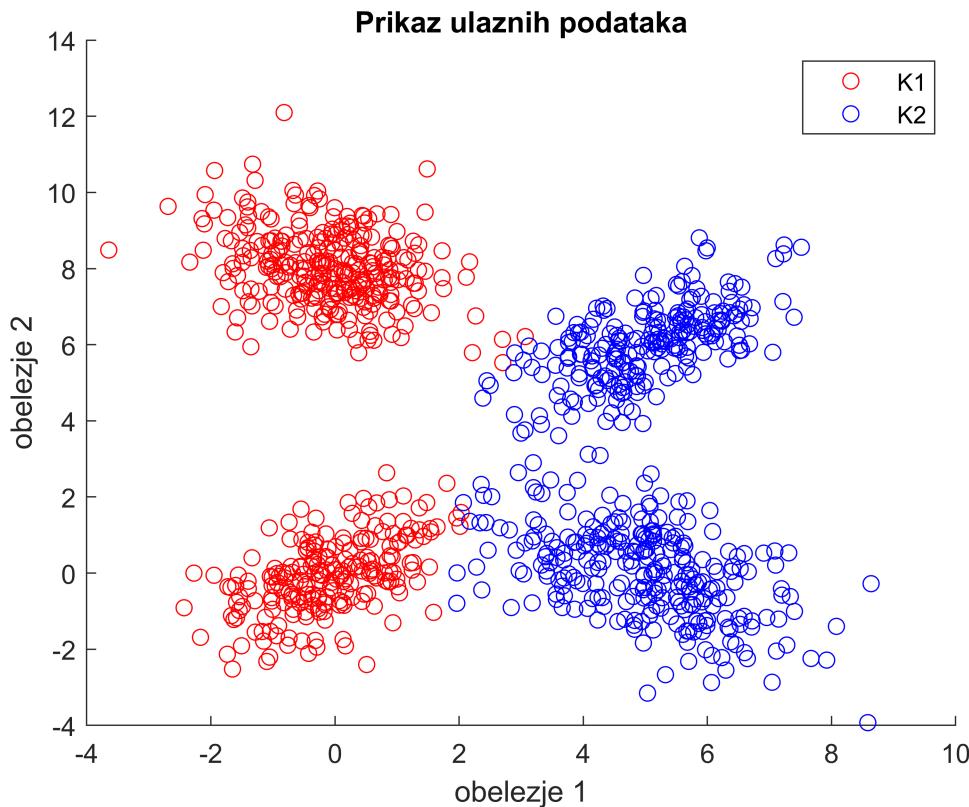
K1 = (pom1<0.45).*K11+(pom1>=0.45).*K12;

% second distribution
pom2 = rand(N , 1);
K21 = mvnrnd(M21,S21, N);
K22 = mvnrnd(M22,S22,N);

K2 = (pom2<0.6).*K21 + (pom2>=0.6).*K22;

figure(fig_id)
hold all;
scatter(K1(:,1),K1(:,2), 'ro');
scatter(K2(:,1),K2(:,2), 'bo');
title('Prikaz ulaznih podataka')
legend('K1', 'K2')
xlabel('obelezje 1')
ylabel('obelezje 2')

```



```
fig_id = fig_id+1;
```

Bajesov klasifikator minimalne verovatnoće greske i procena fgv

```
x = -5:0.1:12;
y = -5:0.1:12;

f1 = zeros(length(x), length(y));
f2 = zeros(length(x), length(y));
k = zeros(length(x), length(y));

e1_b = 0;
e2_b = 0;

const11 = 1/(2*pi)/(det(S11)^0.5);
const12 = 1/(2*pi)/(det(S12)^0.5);
const21 = 1/(2*pi)/(det(S21)^0.5);
const22 = 1/(2*pi)/(det(S22)^0.5);

for i = 1:length(x)
    for j = 1:length(y)
        X = [x(i) y(j)]';

        f11 = const11*exp(-0.5*(X-M11)']*inv(S11)*(X-M11));
        f12 = const12*exp(-0.5*(X-M12)']*inv(S12)*(X-M12));
        f21 = const21*exp(-0.5*(X-M21)']*inv(S21)*(X-M21));
        f22 = const22*exp(-0.5*(X-M22)']*inv(S22)*(X-M22));
```

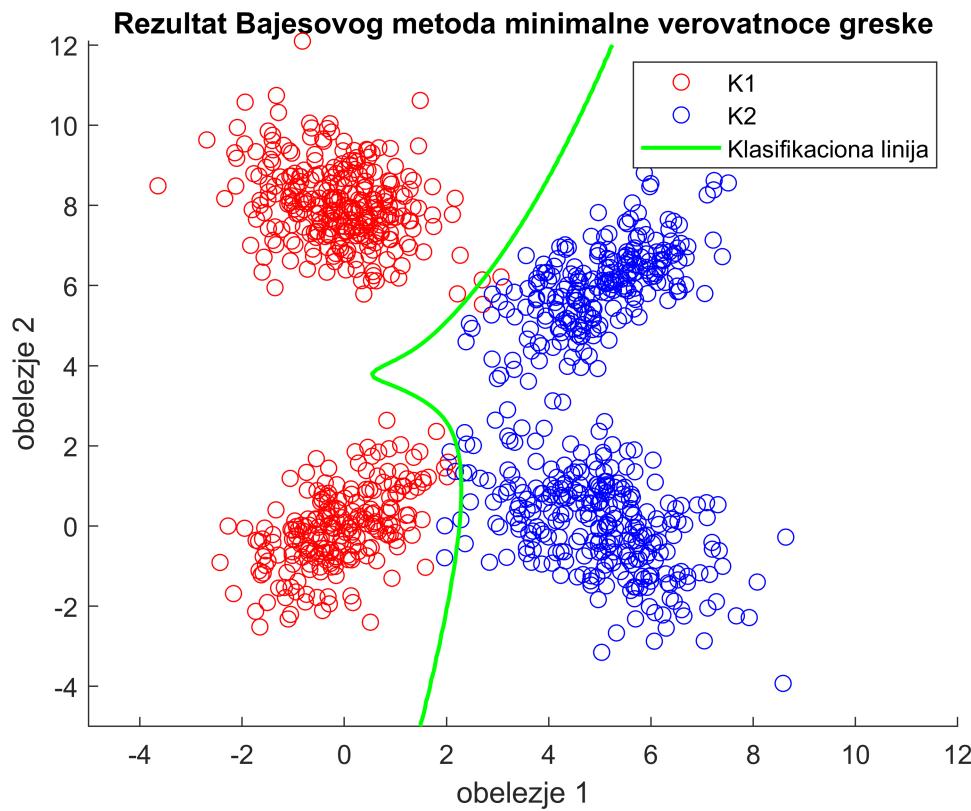
```

f1(i,j) = 0.45*f11+0.55*f12;
f2(i,j) = 0.6*f21+0.4*f22;

k(i,j) = f1(i,j)/f2(i,j);
end
end

figure(fig_id)
hold all;
scatter(K1(:,1),K1(:,2), 'ro');
scatter(K2(:,1),K2(:,2), 'bo');
contour(x,y,k',[1,1], 'g', 'LineWidth',1.5)
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1','K2','Klasifikaciona linija')
title('Rezultat Bajesovog metoda minimalne verovatnoce greske')

```



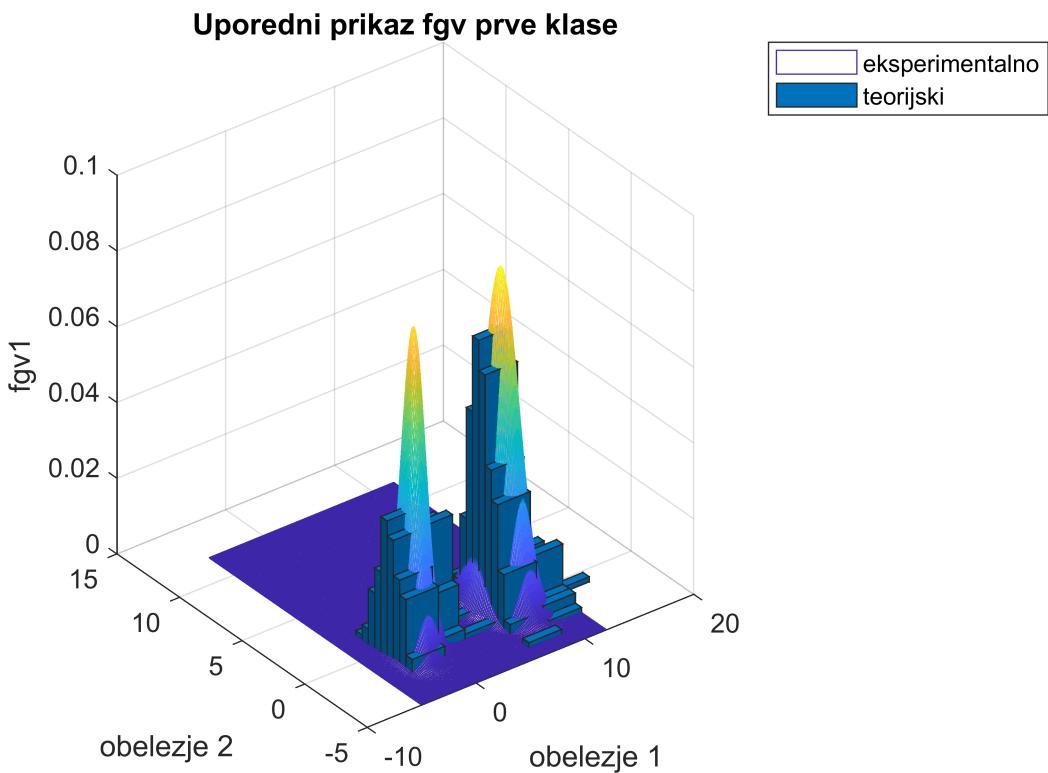
```

fig_id = fig_id+1;

[X,Y] = meshgrid(x,y);
figure(fig_id)
mesh(X,Y,f1, 'FaceAlpha',0.5)
hold on;
histogram2(K1(:,2),K1(:,1), 'Normalization', 'pdf');
xlabel('obelezje 1')
ylabel('obelezje 2')
zlabel('fgv1')
title('Uporedni prikaz fgv prve klase')

```

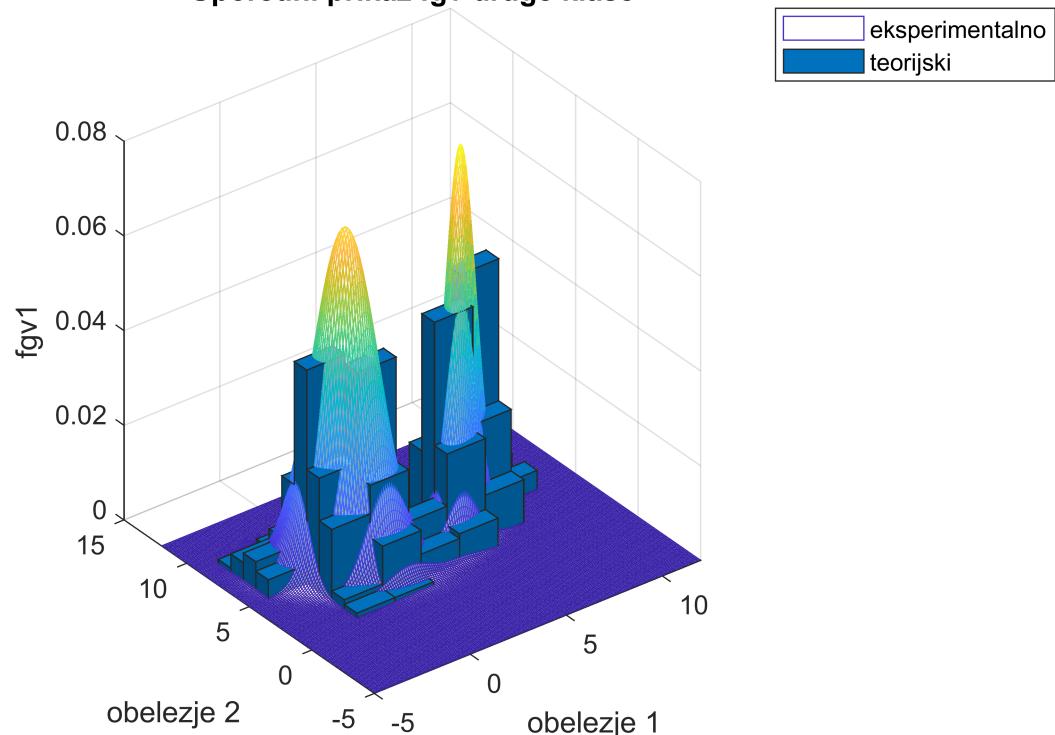
```
legend('eksperimentalno','teorijski')
```



```
fig_id = fig_id+1;

figure(fig_id)
mesh(X,Y,f2,'FaceAlpha',0.5)
hold on;
histogram2(K2(:,2),K2(:,1),'Normalization','pdf');
xlabel('obelezje 1')
ylabel('obelezje 2')
zlabel('fgv1')
title('Uporedni prikaz fgv druge klase')
legend('eksperimentalno','teorijski')
```

Uporedni prikaz fgv druge klase



```
fig_id = fig_id+1;
```

Vidimo da se procena fgv relativno poklapa sa teorijskom ali nije bas najsajnija. Bila bi bolja kada bismo imali vise podataka. Bajesov klasifikator minimalne verovatnoce greske donosi odluku na osnovu aposteriorne verovatnoce, i Bajesove teoreme. On minimizuje verovatnocu ukupne greske podesavanjem optimalnog praga odlucivanja. Greska e2 se racuna u oblasti gde je doneta odluka 1 preko fgv druge klase. Analogno za e1. Sve procene se rade iskljucivo nad obucavajucim skupom.

```
Xs = [K1' K2'];
[e1_b, e2_b] = calc_e_theory(k,1,x,y,f1,f2);
[err1_b, err2_b] = calc_e_exp(Xs, M11, M12, M21,M22,S11,S12, S21,S22);
```

Confusion matrix

| | |
|-----|-----|
| 498 | 2 |
| 4 | 496 |

```
disp_error(e1_b,e2_b,err1_b,err2_b);
```

| | greska prvog reda | greska drugog reda |
|--------------|-------------------|--------------------|
| pravi podaci | 0.004 | 0.008 |
| teorijski | 0.0070182 | 0.0067827 |

```

function [err1, err2] = calc_e_exp(Xs, M11, M12, M21,M22, S11, S12, S21, S22)
const11 = 1/(2*pi)/(det(S11)^0.5);
const12 = 1/(2*pi)/(det(S12)^0.5);
const21 = 1/(2*pi)/(det(S21)^0.5);
const22 = 1/(2*pi)/(det(S22)^0.5);
N = 500;
Y_true = [ones(1,N) ones(1,N)*2];
Y_pred = zeros(size(Y_true));

for i = 1:length(Xs)
    X = Xs(:,i);

    f11 = const11*exp(-0.5*(X-M11) '*inv(S11)*(X-M11));
    f12 = const12*exp(-0.5*(X-M12) '*inv(S12)*(X-M12));
    f21 = const21*exp(-0.5*(X-M21) '*inv(S21)*(X-M21));
    f22 = const22*exp(-0.5*(X-M22) '*inv(S22)*(X-M22));

    f1_curr = 0.45*f11+0.55*f12;
    f2_curr = 0.6*f21+0.4*f22;
    if (f1_curr>f2_curr)
        Y_pred(i) = 1;
    else
        Y_pred(i) = 2;
    end
end

C = confusionmat(Y_true, Y_pred);
disp('Confusion matrix')
disp(C)
%greska prvog tipa
err1 = C(1,2)/sum(C(1,:));
% greska drugog tipa
err2 = C(2,1)/sum(C(2,:));

```

```

function [e1, e2] = calc_e_theory(k, t, x, y, f1, f2)
e1 = 0;
e2 = 0;
for i = 1:length(x)
    for j = 1:length(y)
        if k(i,j)>t % u oblasti L1
            e2 = e2 + 0.1*0.1*f2(i,j);
        else
            e1 = e1+ 0.1*0.1*f1(i,j);
        end
    end
end

```

```

function disp_error(e1, e2, err1,err2)
A = [err1, err2; e1, e2];
T = array2table(A);
T.Properties.VariableNames(1:2) = {'greska prvog reda','greska drugog reda'};
T.Properties.RowNames(1:2) = {'pravi podaci','teorijski'};
disp(T)

```

Greska klasifikacije se racuna nad test skupom. Vidimo da se teorijska i eksperimentalna greska drugog reda poprilično poklapaju ali eksperimentalna greska prvog reda poprilično odstupa od teorijske. Konfuziona matrica nam pokazuje da nas klasifikator radi dobro posao sa TPR = 99.6%, TNR = 99.2% i ACC = 99.4%.

Bajesov klasifikator minimalne cene

Ovaj klasifikator nam daje mogucnost da vise kaznimo gresku pri klasifikaciji jedne klase u odnosu na drugu. Ovo je bitno kada imamo neki problem gde su posledice greske neke klase tragicne, poput analize podataka u medicini kada nam je bitnije da bolesnog pacijenta prepoznamo na vreme nego da zdravom pacijentu kazemo da je zdrav.

U ovu svrhu se definisu cene gde je cena C_{ij} cena koju placamo kada odbirak koji zapravo pripada klasi j smestimo u klasu i . S obzirom da je zahtev zadatka da se vise penalizuje pogresna klasifikacija prve klase postavicom C_{21} na vecu vrednost u odnosu na C_{12} , dok C_{11} i C_{22} mogu biti nula jer oni označavaju da je odbirak dobro klasifikovan.

Ovime se klasifikaciona linija pomera dalje od klase 1 a blize klasi 2.

```

c11 = 0; c22 = 0;
% penalizing K1 misclassification
c12 = 1; c21 = 10;
t = (c12-c22)/(c21-c11);

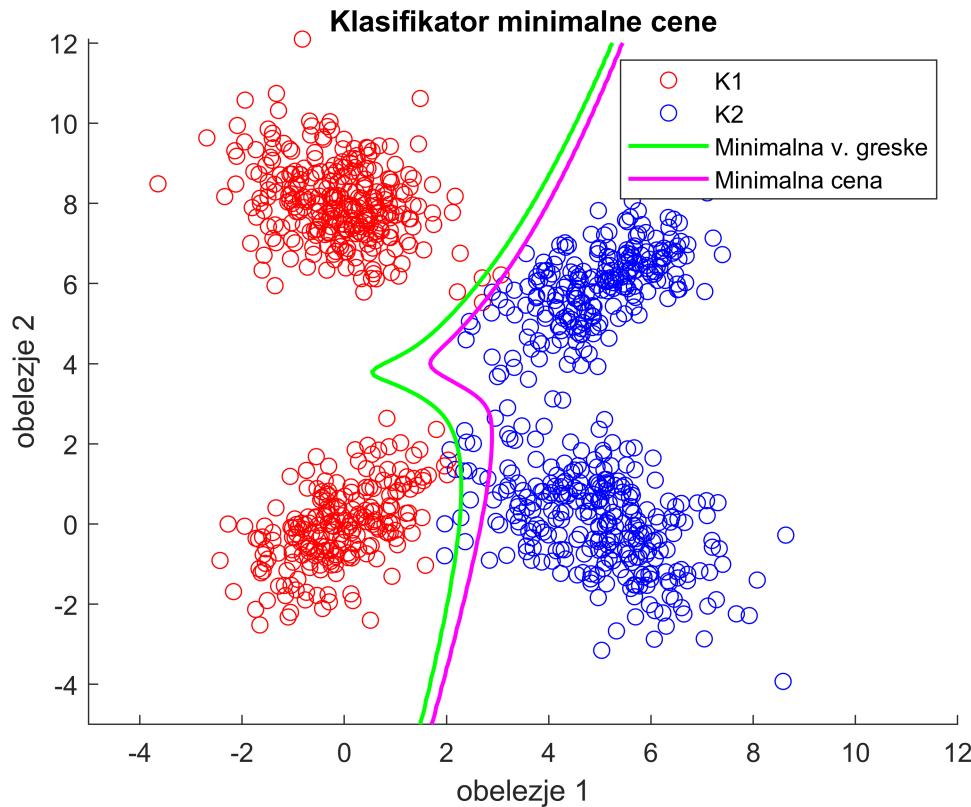
figure(fig_id)
hold all;
scatter(K1(:,1),K1(:,2), 'ro');

```

```

scatter(K2(:,1),K2(:,2), 'bo');
contour(x,y,k',[1,1], 'g', 'LineWidth',1.5)
contour(x,y,k',[t t], 'm', 'LineWidth',1.5);
legend('K1','K2','Minimalna v. greske','Minimalna cena')
xlabel('obelezje 1')
ylabel('obelezje 2')
title('Klasifikator minimalne cene')

```



```
fig_id = fig_id+1;
```

Vidimo da ovaj klasifikator pomera granicu ka klasi 2 u odnos na prosli tako da elementi klase 1 budu bolje klasifikovani na ustrb toga da veci deo elemenata klase 2 bude lose klasifikovan u odnos na malo pre.

```
[e1_c, e2_c] = calc_e_theory(k,t,x,y,f1,f2);
[err1_c, err2_c] = calc_e_exp(Xs, M11, M12, M21,M22,S11,S12, S21,S22);
```

```
Confusion matrix
498    2
 4   496
```

```
disp_error(e1_c,e2_c,err1_c,err2_c);
```

| | greska prvog reda | greska drugog reda |
|--------------|-------------------|--------------------|
| pravi podaci | 0.004 | 0.008 |
| teorijski | 0.0014825 | 0.023017 |

Iz matrice konfuzije se moze primetiti da je greska drugog tipa mnogo porasla dok je greska prvog tipa postala 0. Menjanjem C21 mogu se postici razlicite vrednosti ovih gresaka. Sto je vece C21 to ce biti manja greska prvog tipa i veca gresa drugog tipa i obrnuto.

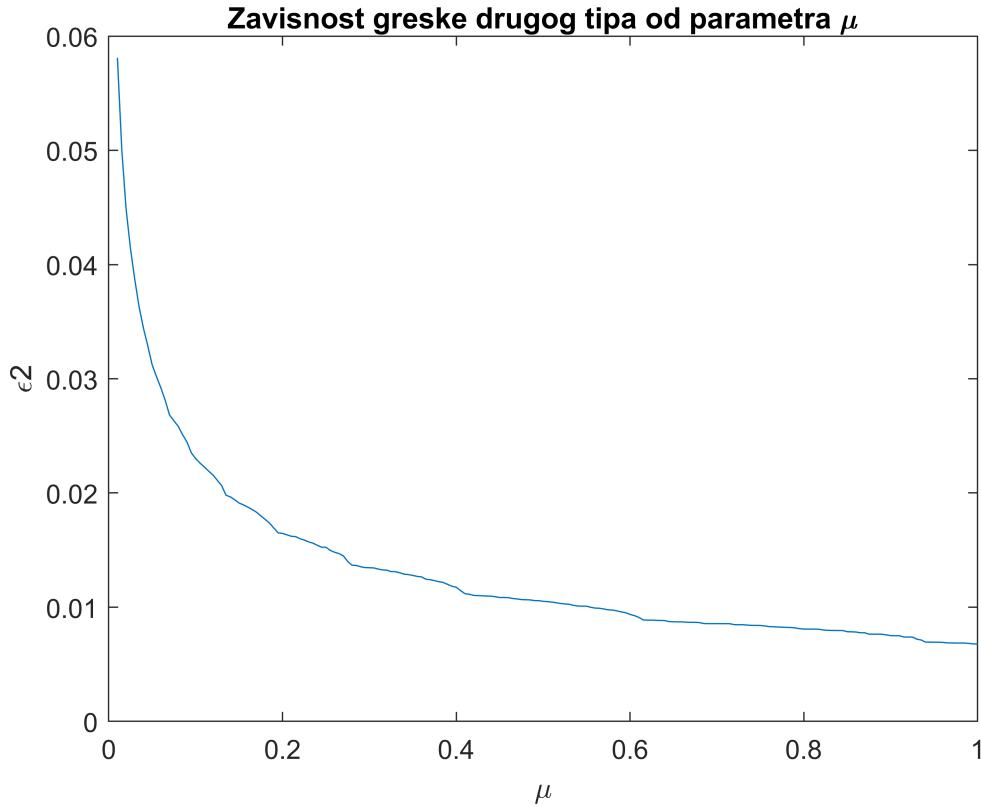
Neuman Pearson-ov klasifikator

Ovaj klasifikator se koristi kada nam je bitno da obe greske budu razumne, da se ne bi desilo da smanjivanjem greske prvog tipa kao malo pre previse pogorsamo gresku drugog tipa. Resenje ovoga je da gresku drugog tipa postavimo na konstantnu vrednost i uporedo smanjujemo gresku prvog tipa. Ovime smo osigurali da greska drugog tipa nece otici iznad razumne vrednosti.

Faktor mu koji figurise kao parametar, odredujemo tako sto konstantu kojoj je jednaka greska drugog tipa stavljamo na vrednost bajesove greske, zato sto je bajesova greska najbolja procena ukupne greske a greska drugog tipa ne moze biti veca od ukupne pa je to dovoljno dobra procena.

```
br = 0;
mii = 0.01:0.005:1;
for mi = mii
    br = br+1;
    [Eps1(br),Eps2(br)] = calc_e_theory(k,mi,x,y,f1,f2);

end
figure(fig_id)
plot(mii,Eps2);
xlabel('\mu');
ylabel('\epsilon_2')
title('Zavisnost greske drugog tipa od parametra \mu')
```

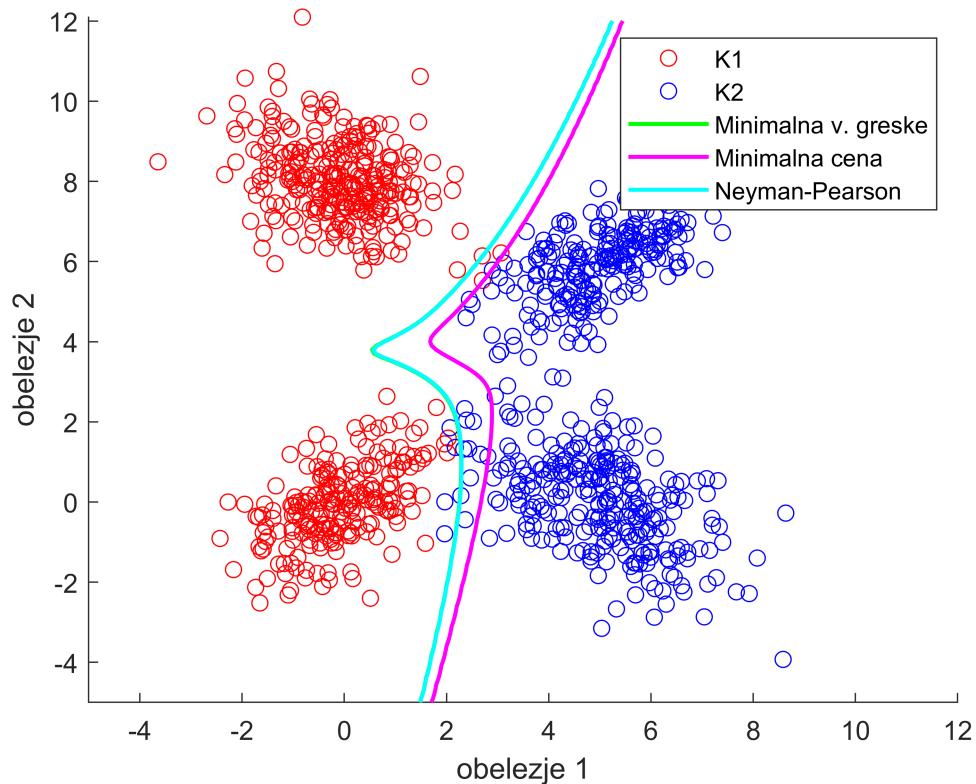


```
fig_id = fig_id+1;
Eps = (e1_b+e2_b)/2;
```

```
id = find(Eps2<=Eps);
e1_n = Eps1(id(1));
e2_n = Eps2(id(1));
mu = mii(id(1));
disp(['Izabrano mu: ' num2str(mu) ', a greska drugog reda za to mu: ' num2str(Eps2(id(1)))]);
```

Izabrano mu: 0.965, a greska drugog reda za to mu: 0.0068751

```
figure(fig_id)
hold all;
scatter(K1(:,1),K1(:,2), 'ro');
scatter(K2(:,1),K2(:,2), 'bo');
contour(x,y,k',[1,1], 'g', 'LineWidth',1.5)
contour(x,y,k',[t t], 'm', 'LineWidth',1.5);
contour(x,y,k',[mu, mu], 'c', 'LineWidth',1.5);
legend('K1','K2','Minimalna v. greske','Minimalna cena', 'Neyman-Pearson')
xlabel('obelezje 1')
ylabel('obelezje 2')
```



```
fig_id = fig_id+1;
```

Vidimo da Nayman-Pearsonov-a klasifikaciona linija je ostala skoro ista kao kod Bajesa zato sto Bajes vec dovoljno dobro klasificuje prvu klasu i ako bismo smanjili ovu gresku postalo bi pogubno po drugu klasu cime se resava problem od malo pre.

```
[err1_n, err2_n] = calc_e_exp(Xs, M11, M12, M21,M22,S11,S12, S21,S22);
```

Confusion matrix

| | |
|-----|-----|
| 498 | 2 |
| 4 | 496 |

```
disp_error(e1_n,e2_n,err1_n,err2_n);
```

| | greska prvog reda | greska drugog reda |
|--------------|-------------------|--------------------|
| pravi podaci | 0.004 | 0.008 |
| teorijski | 0.006927 | 0.0068751 |

Ocekivano, teorijska greska drugog reda je jednaka konstanti, a eksperimentalne greske ostaju iste kao kod bajesa zato sto nije moguce postici bolji rezultat od toga za ovaj set podataka.

Wald-ov sekvencijalni klasifikator

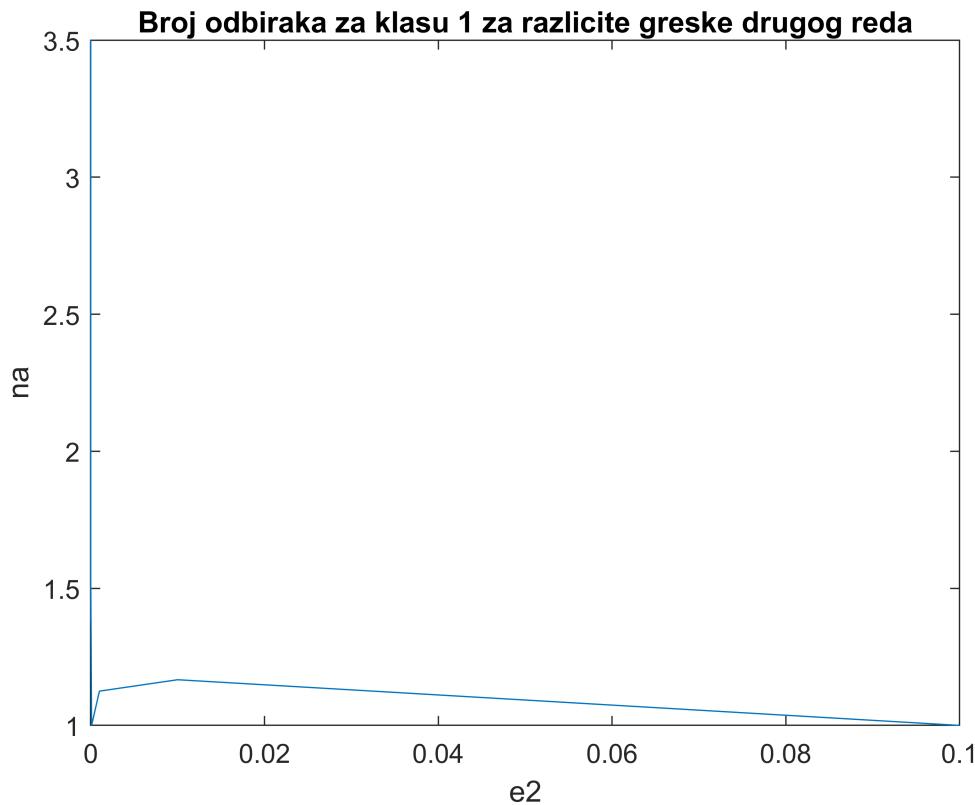
```
M = 15;
```

```

na = zeros(1,M);
nb = zeros(1,M);
eps1 = 0.07;
eps2 = 0.07;
eps = 10.^([-15:-1]);
for i = 1:M
    [na(i), nb(i)] = wald(eps1,eps(i), M11, M12, M21, M22, S11,S12,S21,S22);
end

figure(fig_id)
plot(eps,na)
fig_id = fig_id+1;
title('Broj odbiraka za klasu 1 za razlicite greske drugog reda')
xlabel('e2')
ylabel('na')

```



```

figure(fig_id)
plot(eps,nb)
title('Broj odbiraka za klasu 2 za razlicite greske drugog reda')
xlabel('e2')
ylabel('nb')

```



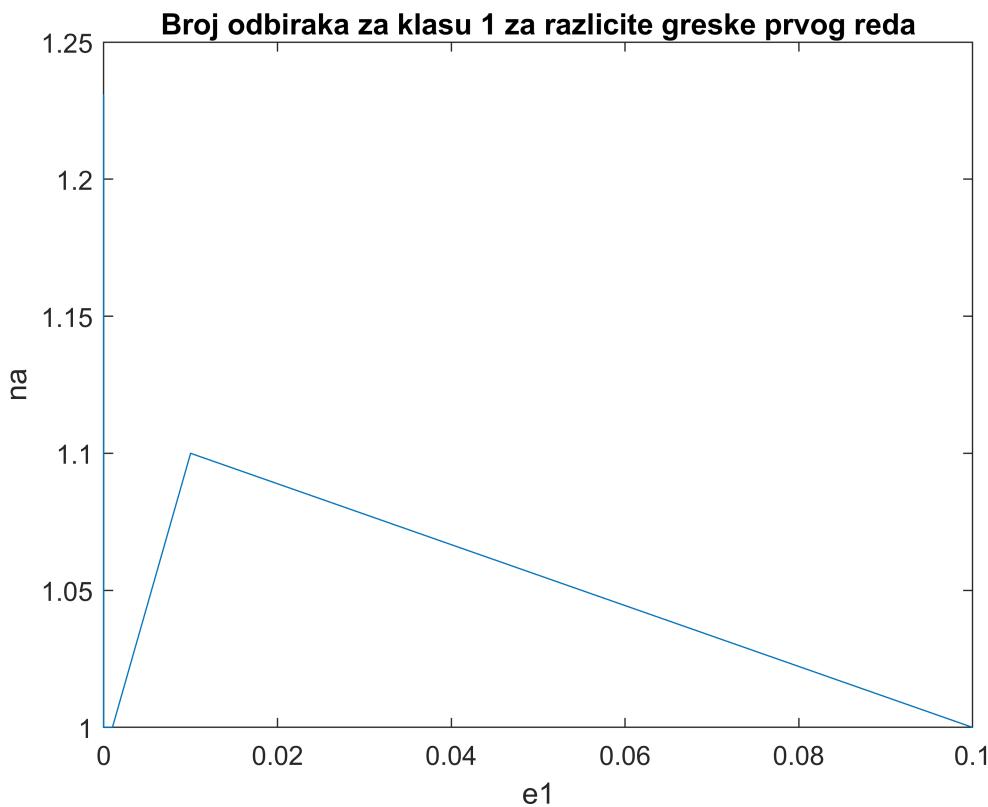
```

fig_id = fig_id+1;

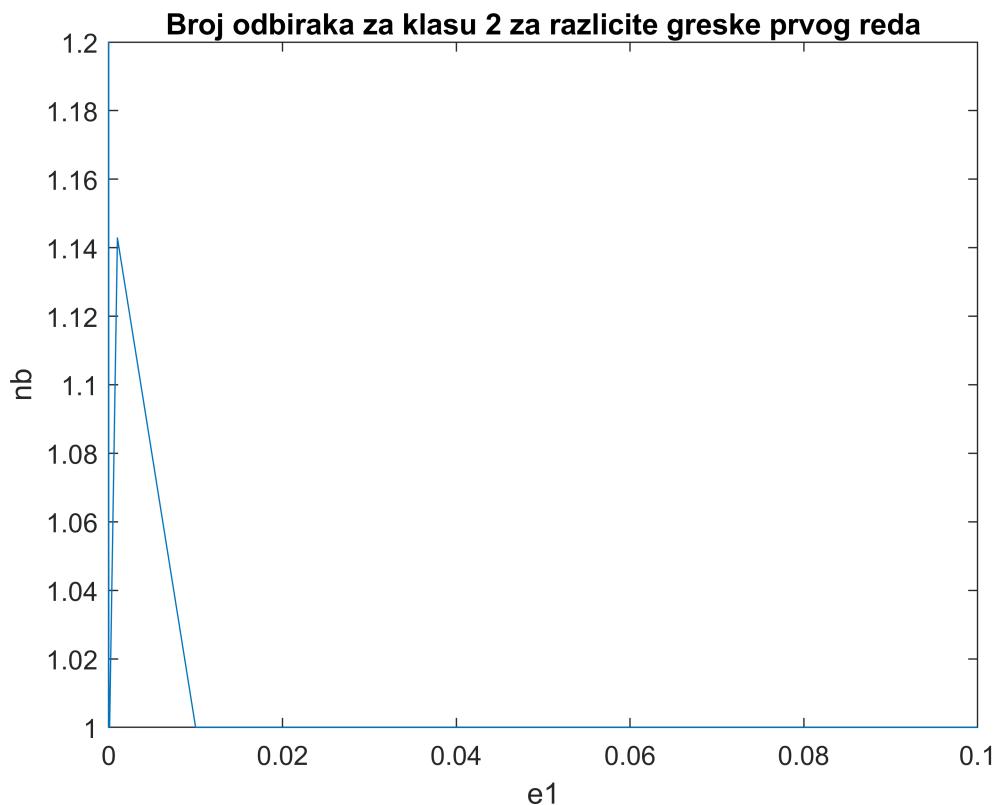
for i = 1:M
    [na(i), nb(i)] = wald(eps(i),eps2, M11, M12, M21, M22, S11,S12,S21,S22);
end

figure(fig_id)
plot(eps,na)
fig_id = fig_id+1;
title('Broj odbiraka za klasu 1 za razlicite greske prvog reda')
xlabel('e1')
ylabel('na')

```



```
figure(fig_id)
plot(eps,nb)
title('Broj odbiraka za klasu 2 za razlicite greske prvog reda')
xlabel('e1')
ylabel('nb')
```



```
fig_id = fig_id+1;
```

Zadatak 3

1. Generisati tri klase dvodimenzionalnih oblika. Izabratи funkciju gustine verovatnoće oblika tako da klase budu linearno separabilne.

- Za tako generisane oblike izvšiti projektovanje linearog klasifikatora jednom od tri iterativne procedure. Rezultate prikazati u obliku matrice konfuzije. Detaljno opisati postupak klasifikacije.
- Ponoviti prethodni postupak korišćenjem metode željenog izlaza. Analizirati uticaj elemenata u matrici željenih izlaza na konačnu formu linearog klasifikatora.

2. Generisati dve klase dvodimenzionalnih oblika koje jesu seperabilne, ali ne linearno, pa isprojektovati kvadratni klasifikator metodom po želji.

Generisanje i prikaz podataka

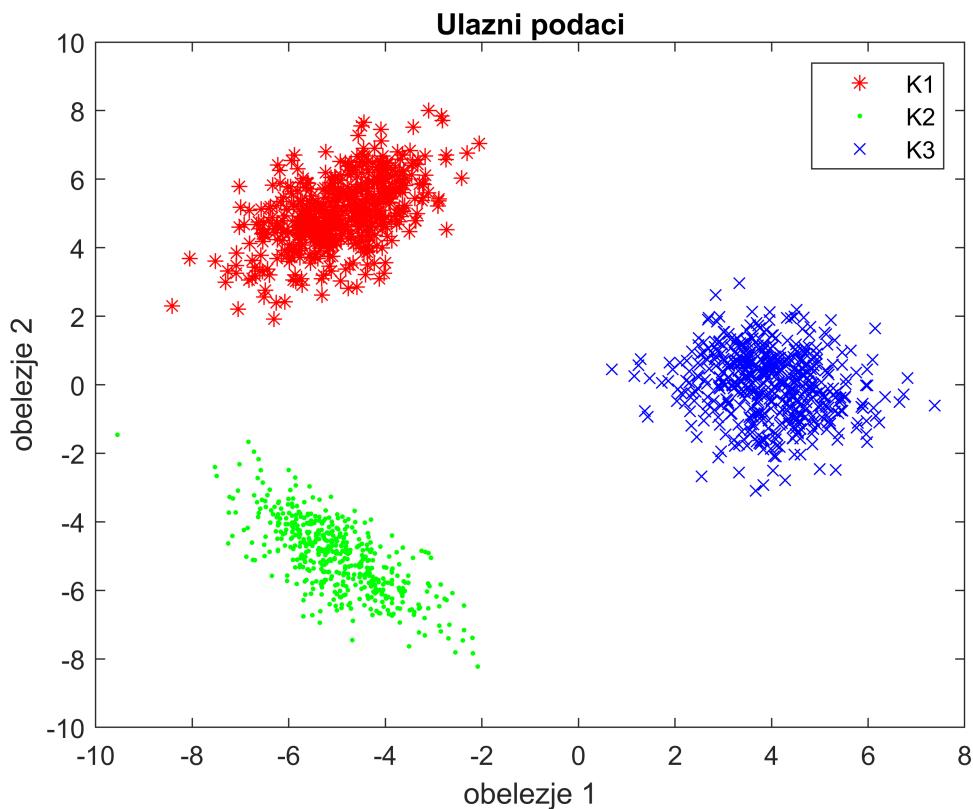
Generisacemo podatke sa Gausovom raspodelom tako da su ocekivanja na dovoljno velikim rastojanjima a kovarijacione matrice dovoljno male da bi bile linearno separabilne.

```
N = 500;

M1 = [-5 5]';
S1 = [1 0.5; 0.5 1];
M2 = [-5 -5]';
S2 = [1 -0.7; -0.7 1];
M3 = [4 0]';
S3 = [1 -0.2; -0.2 1];

K1 = mvnrnd(M1,S1, N)';
K2 = mvnrnd(M2,S2, N)';
K3 = mvnrnd(M3,S3, N);

figure(fig_id)
plot(K1(1,:),K1(2,:),'r*')
hold on;
plot(K2(1,:),K2(2,:),'g.')
hold on;
plot(K3(1,:),K3(2,:),'bx')
hold on;
title('Ulazni podaci')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1','K2','K3')
```



```
fig_id = fig_id+1;
```

Podela podataka na test obucavajuci skup

```
K1 = K1(:, randperm(size(K1, 2)));
K1_test = K1(:,1:150);
K1_train = K1(:,151:end);

K2 = K2(:, randperm(size(K2, 2)));
K2_test = K2(:,1:150);
K2_train = K2(:,151:end);

K3 = K3(:, randperm(size(K3, 2)));
K3_test = K3(:,1:150);
K3_train = K3(:,151:end);
```

Projektovanje optimalnog linearog klasifikatora

Koristicemo hold-out numericki metod, da bi nasa procena bila nepristrasna i da bismo izbegli kompjuterski zahtevno racunanje integrala.

Parametri klase i parametri V i v0 se racunaju nad obucavajucim skupom pomocu formula koje su dobijene tako da minimizuju kriterijumsku funkciju koja je u slucaju optimalnog linearog klasifikatora funkcija verovatnoce greske u slucaju gausovski raspodeljenih oblika. Da bismo izbegli racunanje integrala formiramo prakticno grid u opsegu mogucih y vrednosti skupa koje imamo i nalazimo optimalno v0 za ovaj skup koje ne mora biti (i cesto nije) globalni optimum ali dovoljno dobro radi.

Svi odbirci prve klase za koje je $y_1 > -v_0$ su loše klasifikovani pa treba naci broj ovakvih odbiraka kao i broj odbiraka koji su iz druge klase i vazi $y_2 < -v_0$.

Nakon sto smo nasli parametre V i v_0 za sve vrednosti s , a zatim racunamo gresku nad test skupom za sve vrednosti s . Na kraju, bira se ono s koje minimicuje gresku sracunatu nad test skupom. Na samom kraju, plotuje se izgleda podataka i klasifikacona linija.

```
function [V, v0] = LinearClassifier(K1_test,K1_train, K2_test, K2_train,...  
                                     labell, label2, x1,x2)  
  
M1 = mean(K1_train, 2);  
M2 = mean(K2_train, 2);  
S1 = cov(K1_train');  
S2 = cov(K2_train');  
  
i = 1;  
  
v0_vec = zeros(1,101);  
  
s_vec = 0:0.01:1;  
for s = s_vec  
    V = inv(s*S1+(1-s)*S2)*(M2-M1);  
    y1 = V'*K1_train;  
    y2 = V'*K2_train;  
    Nerr_min = 10^6;  
    vmin = 0;  
    y=sort([y1, y2]);
```

```

for j=1:length(y)-1
    v0=(y(j)+y(j+1))/2;
    Nerr = sum(y1>-v0) + sum(y2<-v0);
    if(Nerr<Nerr_min)
        Nerr_min = Nerr;
        vmin = v0;
    end
end

v0_vec(i) = vmin;
i = i+1;
end
Neps = 10^6;
for i = 1:length(s_vec)
    V = inv(s_vec(i)*S1+(1-s_vec(i))*S2)*(M2-M1);
    y1_curr = V'*K1_test;
    y2_curr = V'*K2_test;
    Neps_curr = sum(y1_curr>-v0_vec(i)) + sum(y2_curr<-v0_vec(i));

    if(Neps_curr<Neps)
        Neps = Neps_curr;
        V_curr = V;
        v0 = v0_vec(i);
    end
end
V = V_curr;
[X1,X2] = meshgrid(x1,x2);
h = V(1)*X1+V(2)*X2+v0;

figure()
p1 = plot(K1_test(1,:), K1_test(2,:), 'b*');
hold on;
p2 = plot(K1_train(1,:), K1_train(2,:), 'b*');
hold on;
plot(K2_test(1,:), K2_test(2,:), 'rx');
hold on;
plot(K2_train(1,:), K2_train(2,:), 'rx');

```

```

plot(K2_train(1,:), K2_train(2,:), 'rx');
hold on;
contour(x1,x2,h,[0,0], 'k');
legend(['K' num2str(label1) 'test'], ['K' num2str(label1) 'train'],...
['K' num2str(label2) 'test'], ['K' num2str(label2) 'train'],...
'Classification line')
title(['Linear Classifier for labels ' num2str(label1) ' and ' num2str(label2)])

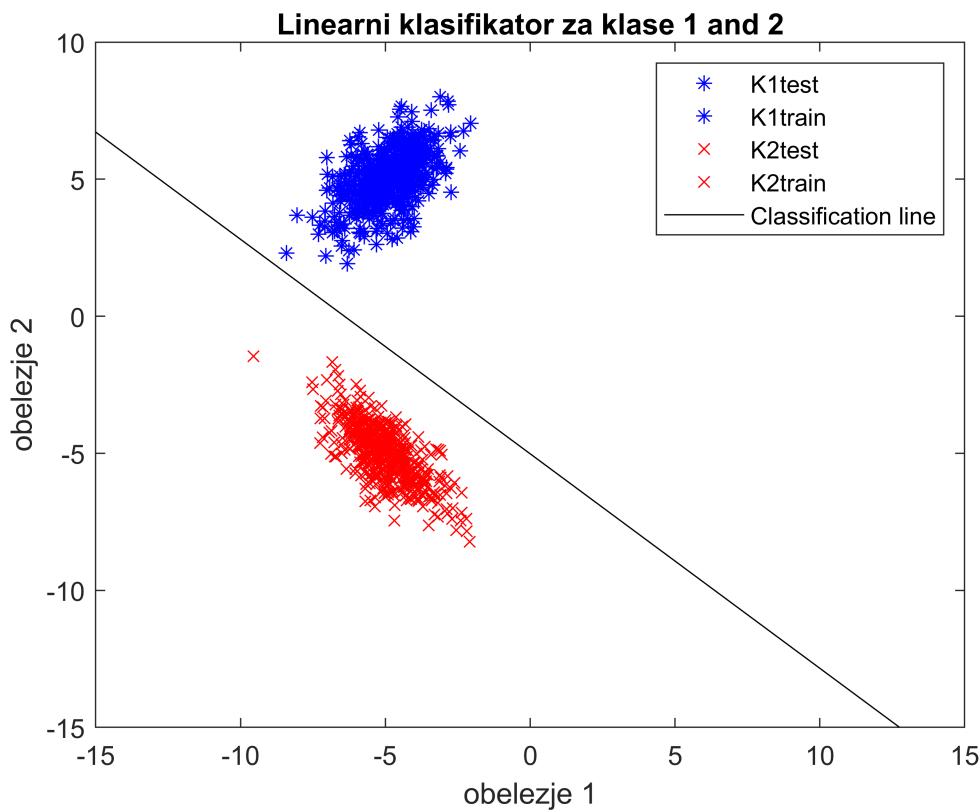
```

Ovaj klasifikator napravljen je za slučaj dve klase. S obzirom da mi imamo tri klase koristicemo deo po deo linearni klasifikator koji formira klasifikacione linije za svake dve klase a zatim klasificuje odbirke tako što ih propusti kroz svaki od klasifikatora (u ovom slučaju ih je 3 - između 1 i 2, 2 i 3, 3 i 1) i klasificuje odbirak u onu klasu koja je dobila najviše glasova za sta se koristi modus ta tri broja.

```

x_grid = -15:0.1:15;
y_grid = -15:0.1:10;
[V12,v012] = LinearClassifier(K1_test,K1_train, K2_test, K2_train,1,2,x_grid, y_grid,fig_id);

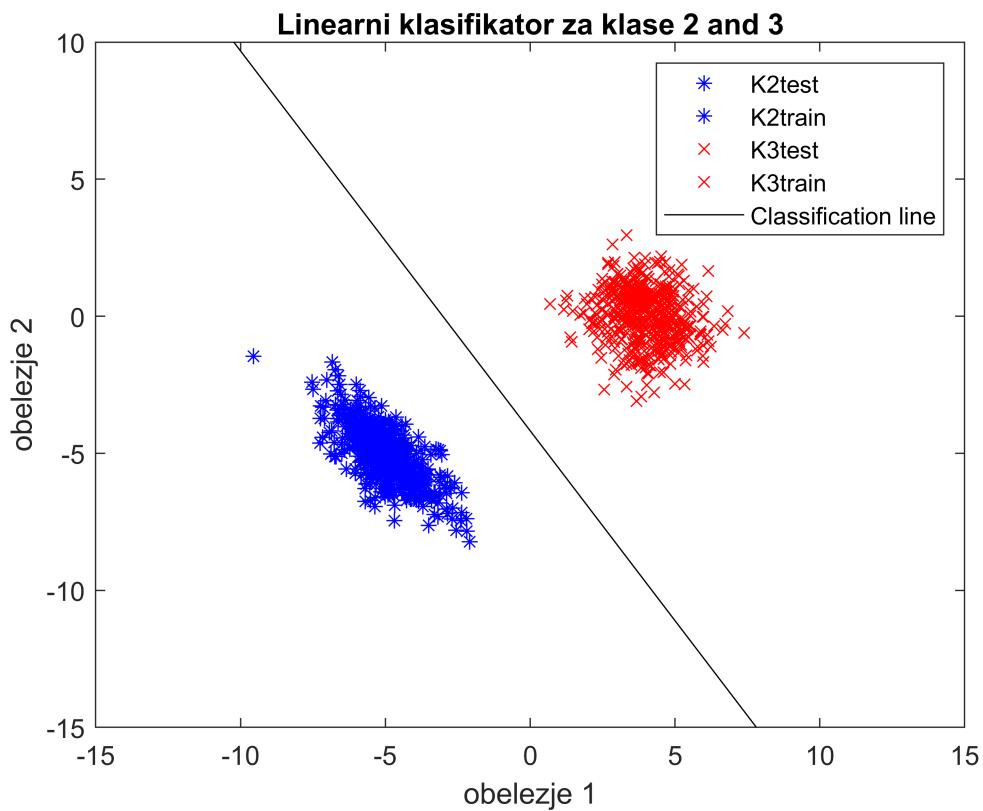
```



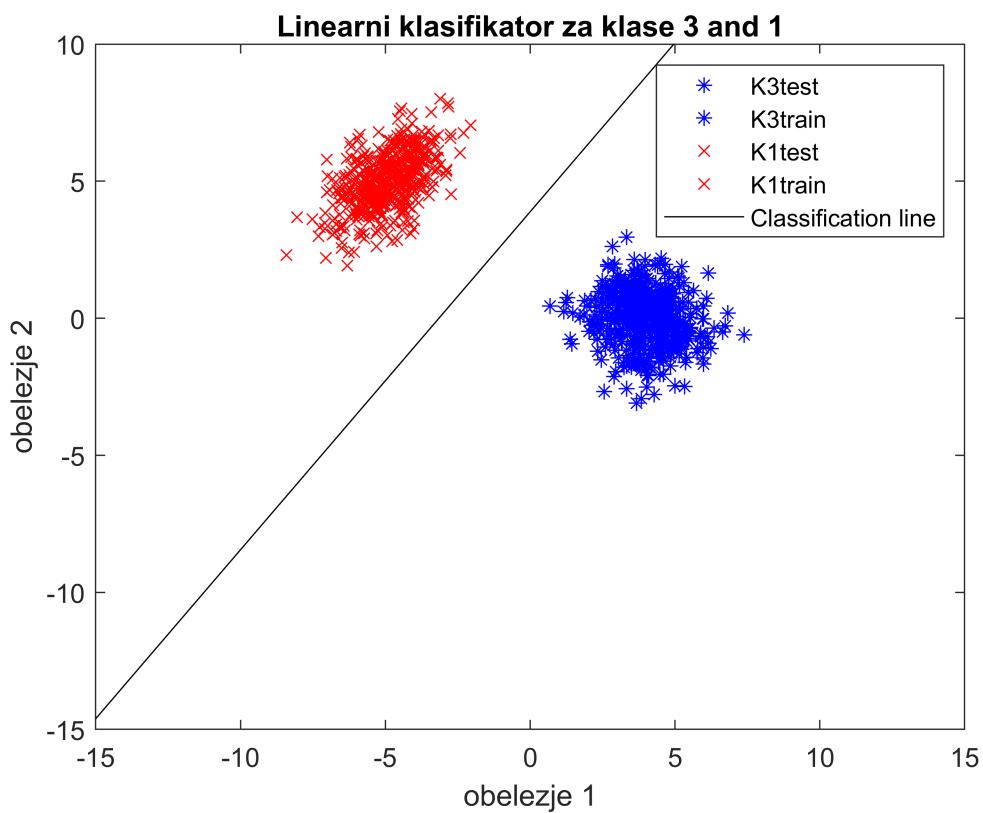
```

fig_id = fig_id+1;
[V23,v023] = LinearClassifier(K2_test,K2_train, K3_test, K3_train,2,3,x_grid,y_grid,fig_id);

```



```
fig_id = fig_id+1;
[V31,v031] = LinearClassifier(K3_test,K3_train, K1_test, K1_train,3,1,x_grid,y_grid,fig_id);
```

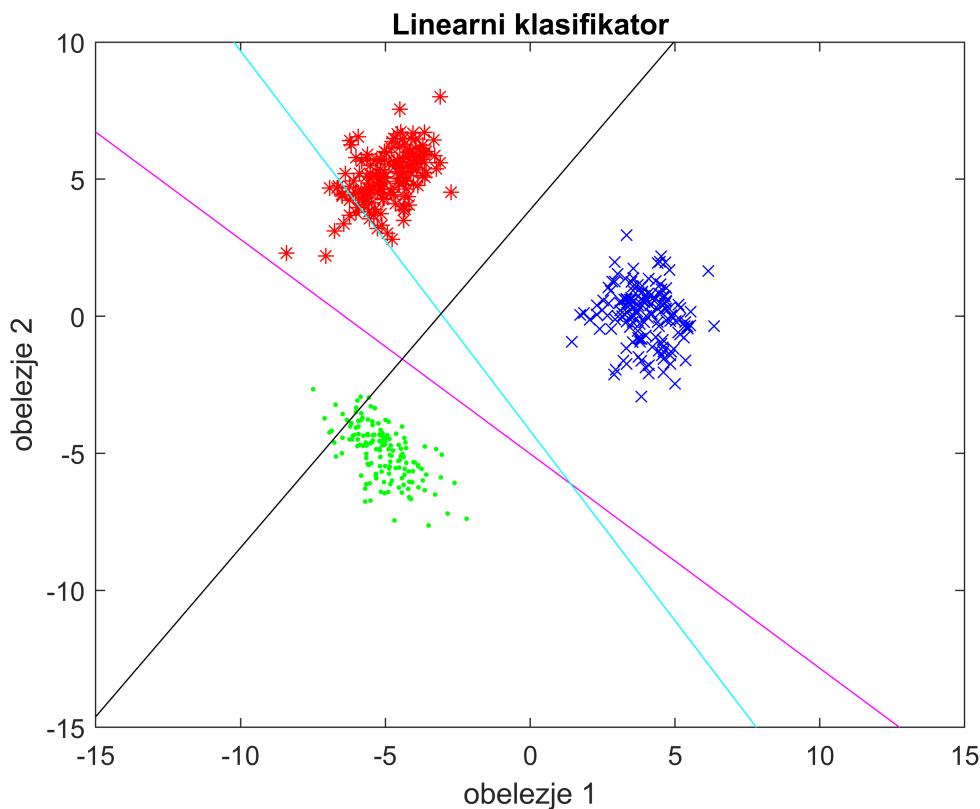


```
fig_id = fig_id +1;
```

```
K_test = [K1_test, K2_test, K3_test];
y1 = zeros(1,length(K_test));
y2 = zeros(1,length(K_test));
y3 = zeros(1,length(K_test));
y_pred = zeros(1,length(K_test));
y_true = [ones(1,150), 2*ones(1,150), 3*ones(1,150)];
cond1 = V12'*K_test+v012>0;
y1(cond1) = 2;
y1(~cond1) = 1;
cond2 = V23'*K_test+v023>0;
y2(cond2) = 3;
y2(~cond2) = 2;
cond3 = V31'*K_test+v031>0;
y3(cond3) = 1;
y3(~cond3) = 3;
```

```
K = [K1, K2, K3];
[X1,X2] = meshgrid(x_grid,y_grid);
h12 = V12(1)*X1+V12(2)*X2+v012;
h23 = V23(1)*X1+V23(2)*X2+v023;
h31 = V31(1)*X1+V31(2)*X2+v031;

y_pred = mode([y1;y2;y3]);
figure(fig_id)
plot(K_test(1,y_pred==1),K_test(2,y_pred==1), 'r*')
hold on;
plot(K_test(1,y_pred==2),K_test(2,y_pred==2), 'g.')
hold on;
plot(K_test(1,y_pred==3),K_test(2,y_pred==3), 'bx')
hold on;
contour(x_grid,y_grid,h12,[0,0], 'm');
hold on;
contour(x_grid,y_grid,h23,[0,0], 'c');
hold on;
contour(x_grid,y_grid,h31,[0,0], 'k');
hold on;
no_mans_land = (y1~=y2 & y2~=y3 & y3~=y1);
title('Linearni klasifikator')
xlabel('obelezje 1')
ylabel('obelezje 2')
```



```
fig_id = fig_id+1;
disp('Broj elemenata u nicioj zemlji')
```

Broj elemenata u nicioj zemlji

```
disp(sum(no_mans_land))
```

0

U opstem slučaju kod ovakvog klasifikatora je moguce dobiti deo ravni za koji nije moguce jednoznačno odrediti kojoj klasi pripada. S obizrom da su nasi ulazni podaci skroz linearne separabilni, ni jedan od podataka ne pripada ovom delu ravni, a konfuziona matrica je dijagonalna jer su svi odbirci potpuno svrstani u svoje klase.

```
C = confusionmat(y_true, y_pred);
disp(C)
```

| | | |
|-----|-----|-----|
| 150 | 0 | 0 |
| 0 | 150 | 0 |
| 0 | 0 | 150 |

Metod zeljenog izlaza

Odbirci prve klase se negiraju a odbirci druge klase ostaju isti cime dobijamo novi ulaz u naš model.

Postavljamo kriterijumsku funkciju koju minimizujemo cime dobijamo koeficijent kojim treba pomnoziti novi ulaz modela da bi sto vise odbiraka pomnozeno tezinom bilo pozitivno. Ta tezina kojom mnozimo odbirke sastoji se iz parametara V i v_0 koriscenih kod linearne klasifikatora a uslov pozitivnosti zamjenjuje uslov znaka kojim

se klasifikuju odbirci u slučaju linearog klasifikatora. Tezina se nalazi metodom pseudoinverzije. Kriterijumska funkcija koja se minimizuje je srednje kvadratna greska.

Vektor Gama nam omogucava da nekim odbircima damo vise znacaja. Mnozenje svih elemenata Gama nekom konstantnom nece nista promeniti jer idalje svi elementi imaju jednak znacaj. Postavljanjem Gama na vecu vrednost za neku klasu pomera klasifikacionu liniju dalje od odbiraka te klase jer time govorimo klasifikatoru da nam je bitnije da ta klasa bude tacno svrstana nego ostale. Takodje, mozemo za neke konkretne odbirke da postavimo Gama na vecu vrednost, na primer ako imamo odbirke koji se nalaze previse blizu granice klasifikacije, mozemo njima dati vecu tezinu jer ako uspemo njih da odvojimo uspecemo i ostale cime pomazemo klasifikatoru.

```
[function [V,v0] = DesiredOutput(K1, K2, G)
N = 350;
Z1 = [-ones(1,N);-K1];
Z2 = [ones(1,N); K2];
Z = [Z1, Z2];
W = inv(Z'*Z')*Z*G;
v0 = W(1);
V = W(2:end);
```

Gama = ones(2N,1)

Prikaz klasifikatora

```
G = ones(2*350, 1);
[V12,v012] = DesiredOutput(K1_train,K2_train,G);
[V23,v023] = DesiredOutput(K2_train, K3_train,G);
[V31,v031] = DesiredOutput(K3_train, K1_train,G);
```

Racunanje vrednosti izlaza:

```
K_test = [K1_test, K2_test, K3_test];
y1 = zeros(1,length(K_test));
y2 = zeros(1,length(K_test));
y3 = zeros(1,length(K_test));

y_true = [ones(1,150), 2*ones(1,150), 3*ones(1,150)];
cond1 = V12'*K_test+v012>0;
y1(cond1) = 2;
y1(~cond1) = 1;
cond2 = V23'*K_test+v023>0;
y2(cond2) = 3;
y2(~cond2) = 2;
cond3 = V31'*K_test+v031>0;
y3(cond3) = 1;
y3(~cond3) = 3;
y_pred = mode([y1;y2;y3]);
```

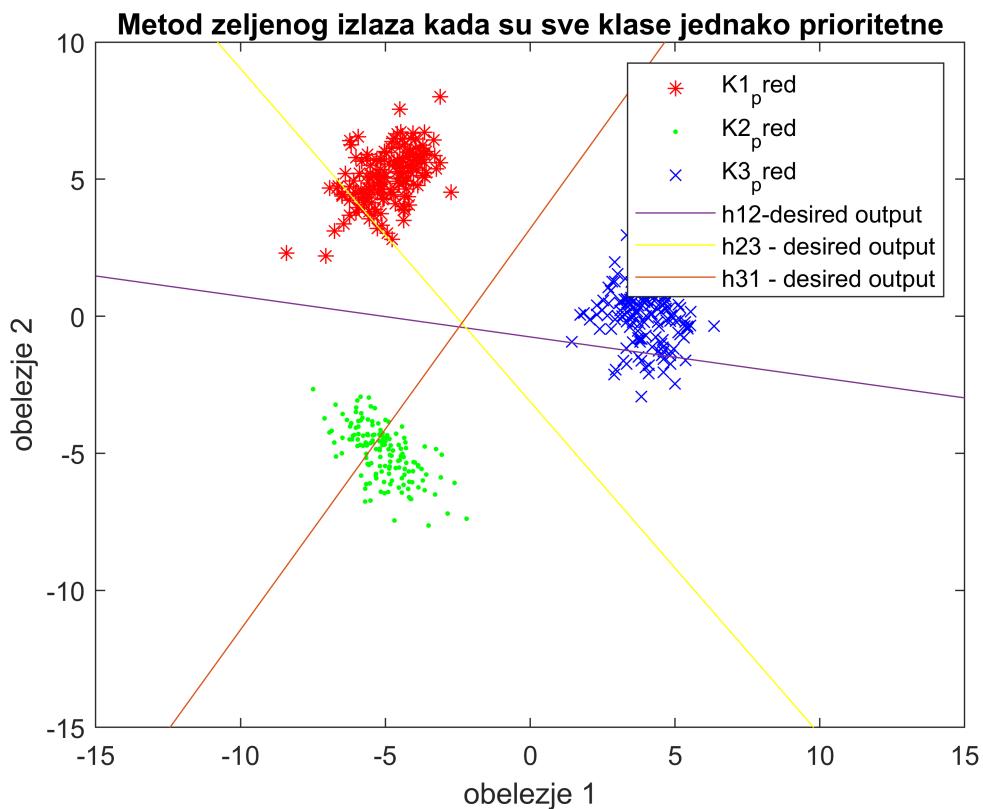
Uporedni prikaz metode zeljenog izlaza i prethodnog metoda

```

K = [K1, K2, K3];
[X1,X2] = meshgrid(x_grid,y_grid);
h12_d = V12(1)*X1+V12(2)*X2+v012;
h23_d = V23(1)*X1+V23(2)*X2+v023;
h31_d = V31(1)*X1+V31(2)*X2+v031;

figure(fig_id)
plot(K_test(1,y_pred==1),K_test(2,y_pred==1), 'r*')
hold on;
plot(K_test(1,y_pred==2),K_test(2,y_pred==2), 'g.')
hold on;
plot(K_test(1,y_pred==3),K_test(2,y_pred==3), 'bx')
hold on;
contour(x_grid,y_grid,h12_d,[0,0], 'EdgeColor', [0.4940 0.1840 0.5560]);
hold on;
contour(x_grid,y_grid,h23_d,[0,0], 'y');
hold on;
contour(x_grid,y_grid,h31_d,[0,0], 'EdgeColor', [0.8500 0.3250 0.0980]);
hold on;
xlabel('obelezje 1')
ylabel('obelezje 2')
title('Metod zeljenog izlaza kada su sve klase jednako prioritetne')
legend('K1_pred','K2_pred','K3_pred', 'h12-desired output','h23 - desired output', 'h31 - desi

```



```
fig_id = fig_id+1;
```

Primecujemo da su klasifikacione linije postavljene tako da su otprilike izmedju klasa.

Konfuziona matrica:

```
C = confusionmat(y_true, y_pred);  
disp(C)
```

```
150      0      0  
 0    150      0  
 0      0    150
```

Svi odbirci su dobro svrstani.

Gama = [ones(N,1),;2*ones(N,1)] za K12, [2* ones(N,1) ;ones(N,1)] za K23, ones(2*N,1) za K13

-> kazemo da su odbirci druge klase 2*bitniji

```
[V12,v012] = DesiredOutput(K1_train,K2_train,[ones(350, 1);2*ones(350, 1)]);  
[V23,v023] = DesiredOutput(K2_train, K3_train,[2*ones(350, 1);ones(350, 1)]);  
[V31,v031] = DesiredOutput(K3_train, K1_train,ones(2*350, 1));
```

Racunanje vrednosti izlaza:

```
K_test = [K1_test, K2_test, K3_test];  
y1 = zeros(1,length(K_test));  
y2 = zeros(1,length(K_test));  
y3 = zeros(1,length(K_test));  
  
y_true = [ones(1,150), 2*ones(1,150), 3*ones(1,150)];  
cond1 = V12'*K_test+v012>0;  
y1(cond1) = 2;  
y1(~cond1) = 1;  
cond2 = V23'*K_test+v023>0;  
y2(cond2) = 3;  
y2(~cond2) = 2;  
cond3 = V31'*K_test+v031>0;  
y3(cond3) = 1;  
y3(~cond3) = 3;  
y_pred = mode([y1;y2;y3]);
```

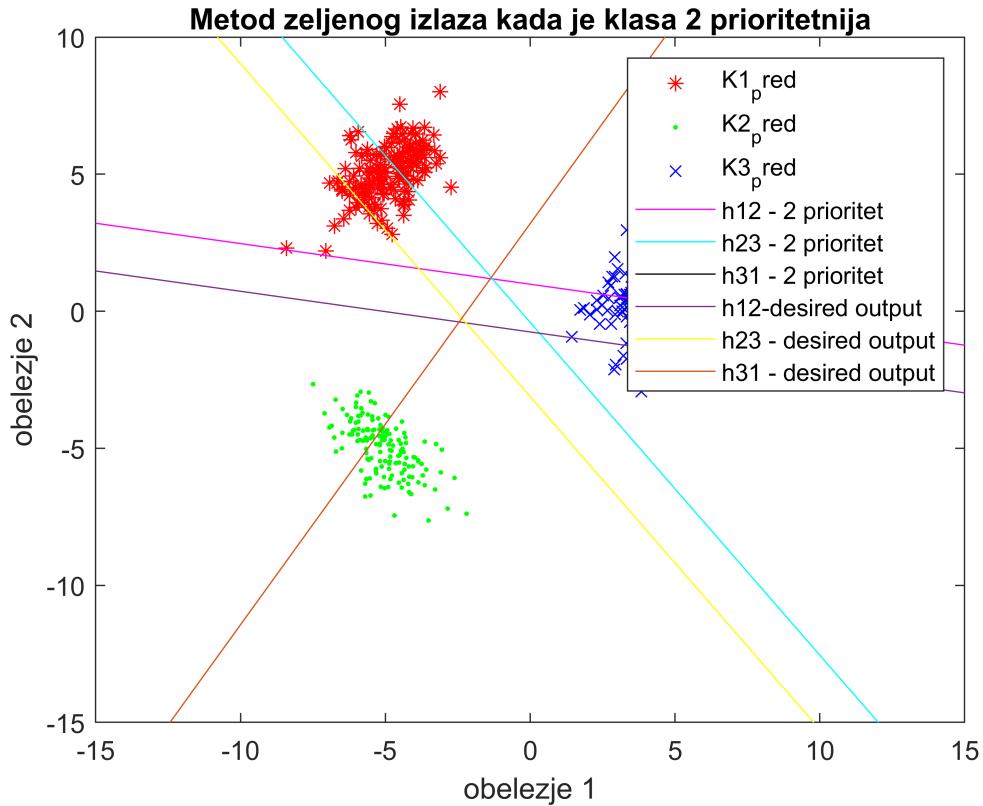
Uporedni prikaz metode zeljenog izlaza za prethodno G gde su svi jednako bitni, i novo G za koje su odbirci klase 2 bitniji.

```
K = [K1, K2, K3];  
[X1,X2] = meshgrid(x_grid,y_grid);  
h12_d_1 = V12(1)*X1+V12(2)*X2+v012;  
h23_d_1 = V23(1)*X1+V23(2)*X2+v023;  
h31_d_1 = V31(1)*X1+V31(2)*X2+v031;
```

```

figure(fig_id)
plot(K_test(1,y_pred==1),K_test(2,y_pred==1), 'r*')
hold on;
plot(K_test(1,y_pred==2),K_test(2,y_pred==2), 'g.')
hold on;
plot(K_test(1,y_pred==3),K_test(2,y_pred==3), 'bx')
hold on;
contour(x_grid,y_grid,h12_d_1,[0,0], 'm');
hold on;
contour(x_grid,y_grid,h23_d_1,[0,0], 'c');
hold on;
contour(x_grid,y_grid,h31_d_1,[0,0], 'k');
hold on;
contour(x_grid,y_grid,h12_d,[0,0], 'EdgeColor', [0.4940 0.1840 0.5560]);
hold on;
contour(x_grid,y_grid,h23_d,[0,0], 'y');
hold on;
contour(x_grid,y_grid,h31_d,[0,0], 'EdgeColor', [0.8500 0.3250 0.0980]);
hold on;
xlabel('obelezje 1')
ylabel('obelezje 2')
title('Metod zeljenog izlaza kada je klasa 2 prioritetnija')
legend('K1_pred','K2_pred','K3_pred','h12 - 2 prioritet','h23 - 2 prioritet','h31 - 2 prioritet')

```



```
fig_id = fig_id+1;
```

Primetimo da su se sada obe klasifikacione linije u cijoj konstrukciji ucestvuje klasa 2 (zelena) pomerile dalje od nje a blize drugim dvema klasama dok je treca klasifikaciona linija ostala ista.

Konfuziona matrica:

```
C = confusionmat(y_true, y_pred);
disp(C)
```

```
150      0      0
 0    150      0
 0      0    150
```

Gama = [1.5*ones(N,1),;ones(N,1)] za K12, ones(2*N,1) za K23, [ones(N,1) ;1.5*ones(N,1)] za K31

-> kazemo da su odbirci prve klase 1,5* bitniji

```
[V12,v012] = DesiredOutput(K1_train,K2_train,[1.5*ones(350, 1);ones(350, 1)]);
[V23,v023] = DesiredOutput(K2_train, K3_train,ones(2*350, 1));
[V31,v031] = DesiredOutput(K3_train, K1_train,[ones(350, 1);1.5*ones(350, 1)]);
```

Racunanje vrednosti izlaza:

```
K_test = [K1_test, K2_test, K3_test];
y1 = zeros(1,length(K_test));
y2 = zeros(1,length(K_test));
y3 = zeros(1,length(K_test));

y_true = [ones(1,150), 2*ones(1,150), 3*ones(1,150)];
cond1 = V12'*K_test+v012>0;
y1(cond1) = 2;
y1(~cond1) = 1;
cond2 = V23'*K_test+v023>0;
y2(cond2) = 3;
y2(~cond2) = 2;
cond3 = V31'*K_test+v031>0;
y3(cond3) = 1;
y3(~cond3) = 3;
y_pred = mode([y1;y2;y3]);
```

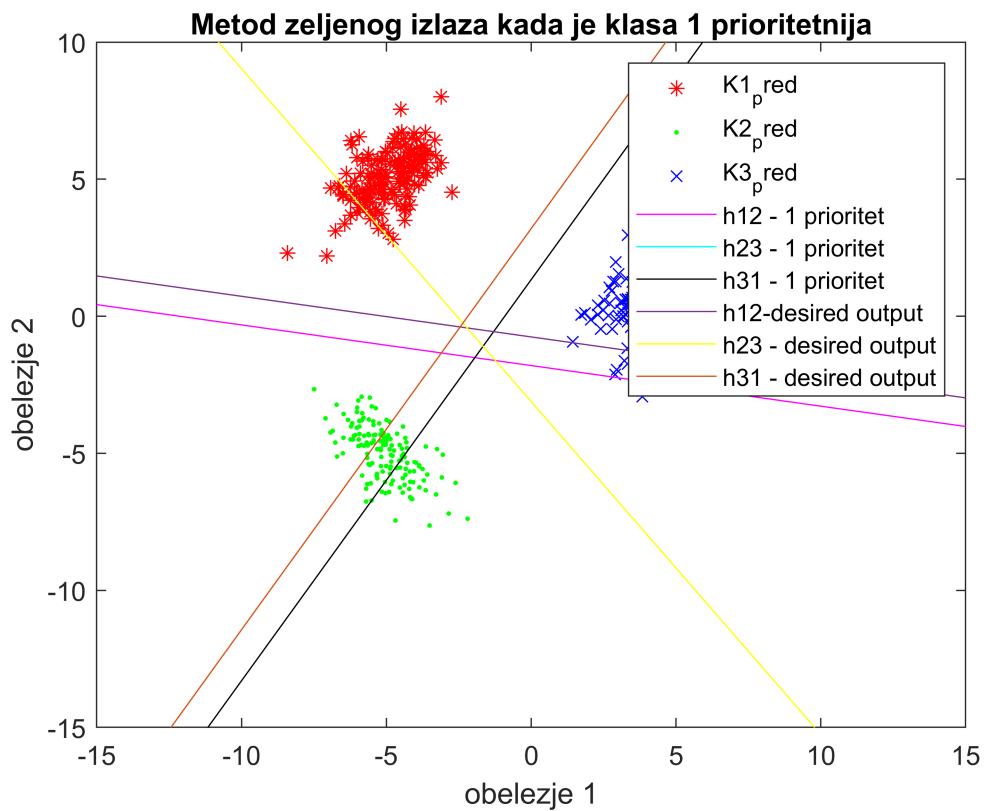
Uporedni prikaz metode zeljenog izlaza za prethodno G gde su svi jednako bitni, i novo G za koje su odbirci klase 2 bitniji.

```
K = [K1, K2, K3];
[X1,X2] = meshgrid(x_grid,y_grid);
h12_d_1 = V12(1)*X1+V12(2)*X2+v012;
h23_d_1 = V23(1)*X1+V23(2)*X2+v023;
h31_d_1 = V31(1)*X1+V31(2)*X2+v031;
```

```

figure(fig_id)
plot(K_test(1,y_pred==1),K_test(2,y_pred==1), 'r*')
hold on;
plot(K_test(1,y_pred==2),K_test(2,y_pred==2), 'g.')
hold on;
plot(K_test(1,y_pred==3),K_test(2,y_pred==3), 'bx')
hold on;
contour(x_grid,y_grid,h12_d_1,[0,0], 'm');
hold on;
contour(x_grid,y_grid,h23_d_1,[0,0], 'c');
hold on;
contour(x_grid,y_grid,h31_d_1,[0,0], 'k');
hold on;
contour(x_grid,y_grid,h12_d,[0,0], 'EdgeColor', [0.4940 0.1840 0.5560]);
hold on;
contour(x_grid,y_grid,h23_d,[0,0], 'y');
hold on;
contour(x_grid,y_grid,h31_d,[0,0], 'EdgeColor', [0.8500 0.3250 0.0980]);
hold on;
xlabel('obelezje 1')
ylabel('obelezje 2')
title('Metod zeljenog izlaza kada je klasa 1 prioritetnija')
legend('K1_pred','K2_pred','K3_pred','h12 - 1 prioritet','h23 - 1 prioritet','h31 - 1 prioritet')

```



```
fig_id = fig_id+1;
```

Primetimo da su se sada obe klasifikacione linije u cijoj konstrukciji ucestvuju klasa 1 (crvena) pomerile dalje od nje a blize drugim dvama klasama dok je druga klasifikaciona linija ostala ista.

Konfuziona matrica:

```
C = confusionmat(y_true, y_pred);
disp(C)
```

```
150      0      0
 0    150      0
 0      0    150
```

Nelinearno separabilni podaci i kvadratni klasifikator

Ideja kod kvadratnog klasifikatora je da od nelinearne funkcije obeležja dobijemo linearu funkciju od novonapravljenih obeležja. Ostali postupak je isti kao kod predjasnjeg klasifikatora. S obzirom da nam ni jedno obeležje nije bitnije koristicemo Gama sa istim prioritetima za sve.

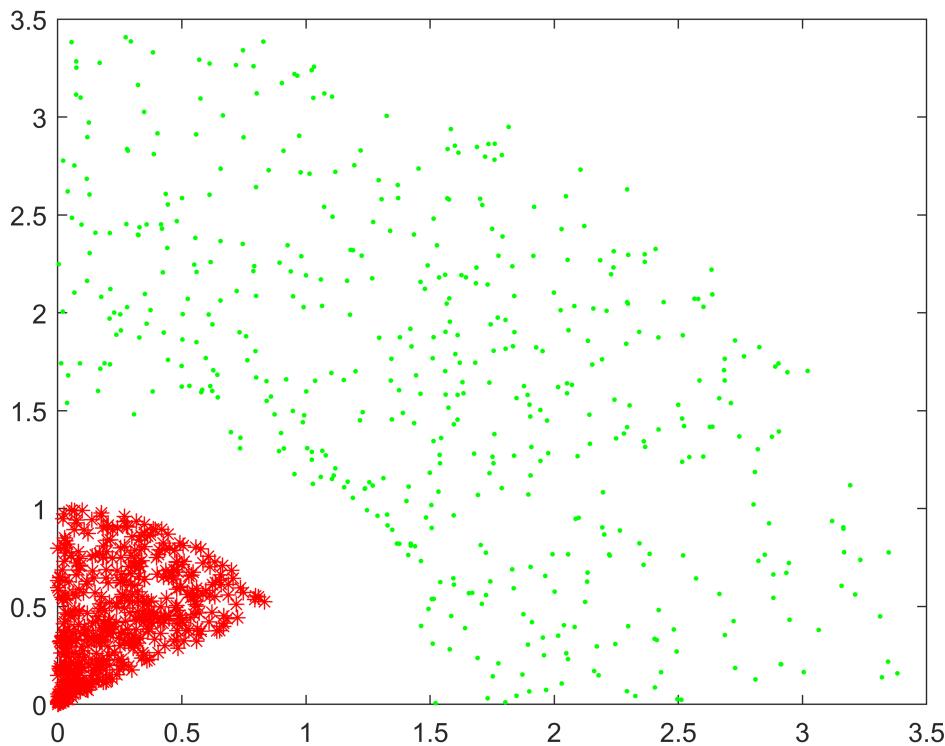
```
N = 500;
x = rand(1,500)*10^-5;
rho = rand(1,500);
theta = rand(1,500)*pi/3;

K1 = [rho.*sin(theta);rho.*cos(theta)];

rho = rand(1,500)*2+1.5;
theta = rand(1,500)*pi/2;

K2 = [rho.*sin(theta);rho.*cos(theta)];

figure()
plot(K1(1,:),K1(2,:),'r*')
hold on;
plot(K2(1,:),K2(2,:),'g.')
hold on;
```



```
K1 = K1(:, randperm(size(K1, 2)));
K1_test = K1(:,1:150);
K1_train = K1(:,151:end);

K2 = K2(:, randperm(size(K2, 2)));
K2_test = K2(:,1:150);
K2_train = K2(:,151:end);
```

```
[V,v0] = QuadraticClassifier(K1_train,K2_train,ones(2*350, 1));
```

Racunanje vrednosti izlaza:

```
K_test = [K1_test, K2_test];

y_true = [ones(1,150), 2*ones(1,150)];
X = [K_test(1,:).^2 ;K_test(2,:).^2;K_test(1,:).*K_test(2,:));K_test];
h = V'*X+v0>0;
y_pred = zeros(1,length(X));
y_pred(h) = 2;
y_pred(~h) = 1;
```

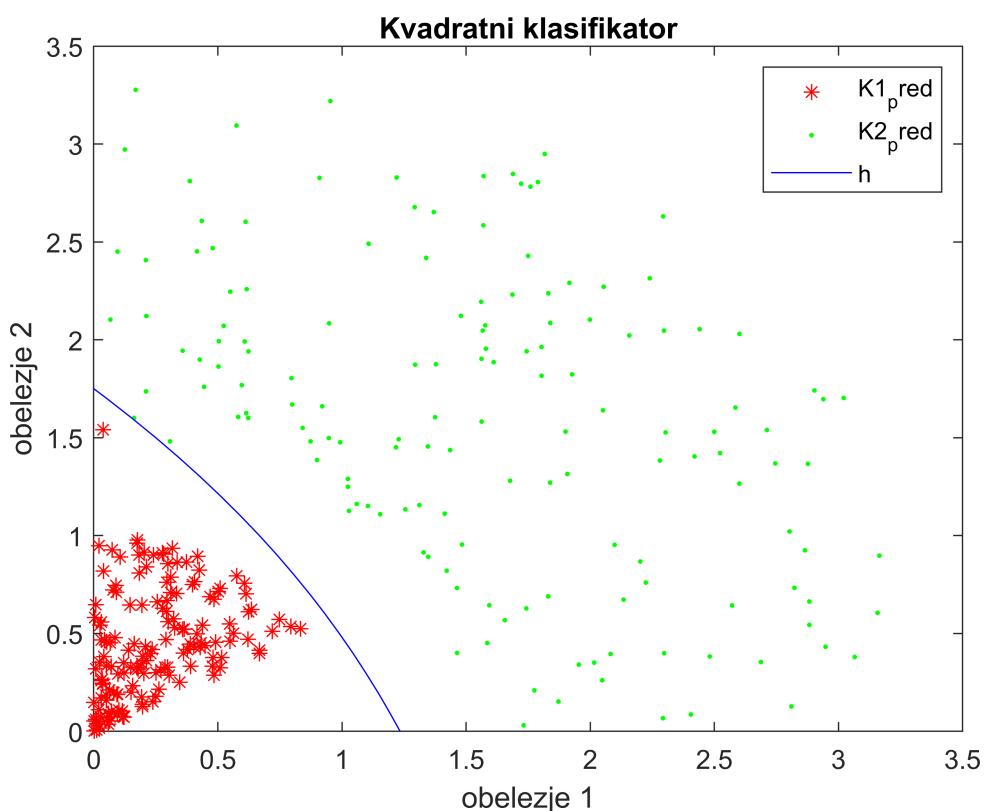
Uporedni prikaz metode zeljenog izlaza za prethodno G gde su svi jednako bitni, i novo G za koje su odbirci klase 2 bitniji.

```

K = [K1, K2];
x_grid = 0:0.1:3.5;
y_grid = 0:0.1:3.5;
[X1,X2] = meshgrid(x_grid,y_grid);
h = v0+V(1)*(X1.^2)+V(2)*(X2.*2)+V(3).*(X1.*X2)+V(4)*(X1)+V(5)*(X2);

figure(fig_id)
plot(K_test(1,y_pred==1),K_test(2,y_pred==1), 'r*')
hold on;
plot(K_test(1,y_pred==2),K_test(2,y_pred==2), 'g.')
hold on;
contour(x_grid,y_grid,h,[0,0], 'b');
xlabel('obelezje 1')
ylabel('obelezje 2')
title('Kvadratni klasifikator')
legend('K1_pred', 'K2_pred', 'h')

```



```
fig_id = fig_id +1;
```

Konfuziona matrica:

```

C = confusionmat(y_true, y_pred);
disp(C)

```

| | |
|-----|-----|
| 150 | 0 |
| 1 | 149 |

Zadatak 4

1. Generisati po $= 500$ dvodimenzionih odbiraka iz četiri klase koje će biti linearno separabilne. Preporuka je da to budu Gausovski raspodeljeni dvodimenzioni oblici. Izabratи jednu od metoda za klasterizaciju (c mean metod, metod kvadratne dekompozicije) i primeniti je na formirane uzorke klase. Izvršiti analizu osetljivosti izabranog algoritma na početnu klasterizaciju kao i srednji broj potrebnih iteracija. Takođe izvršiti analize slučaja kada se apriorno ne poznaje broj klasa.
2. Na odbircima iz prethodne tačke izabratи jednu od metoda klasterizacije (metod maksimalne verodostojnosti ili metod grana i granica) i primeniti je na formirane uzorke klase. Izvršiti analizu osetljivosti izabranog algoritma na početnu klasterizaciju kao i srednji broj potrebnih iteracija. Takođe izvršiti analize slučaja kada se apriorno ne poznaje broj klasa.
3. Generisati po $= 500$ dvodimenzionih odbiraka iz dve klase koje su nelinearno separabilne. Izabratи jednu od metoda za klasterizaciju koje su primenjive za nelinearno separabilne klase (metod kvadratne dekompozicije ili metod maksimalne verodostojnosti) i ponoviti analizu iz prethodnih tačaka.

Podaci 1

C-mean

C-mean algoritam kao kriterijumsku funkciju koristi srednje kvadratno odstupanje odbiraka od srednje vrednosti klastera. Ideja je da svaki odbirak dodelimo klasteru cijoj je srednjoj vrednosti najbliži. Nakon svake iteracije azurira se srednja vrednost klastera i postupak se nastavlja sve dok ne dodjemo do iteracije u kojoj se nije desila ni jedna promena ili dok ne dodjemo do maksimalnog broja iteracija koje zadaje korisnik.

- Broj klastera mora unapred biti poznat. Ako nije, jedini nacin da odsredimo broj klastera je da uradimo elbow metod i odokativno ga procenimo ili se posavetujemo sa nekim ekspertom.
- Pocetna klazterizacija moze da se izvrsi cisto stohasticki i nije nam potrebno apriorno znanje za tacnu klasterizaciju
- Klasifikacione linije su zapravo simetrale duzi koje spajaju srednje vrednosti klastera
- Ne garantuje globalnu stabilnost, cesto moze da se zaglavi u nekom lokalnom minimumu i moze da izdivergira.

```

function [final_clustering, final_l] = Cmeans(K,C,clusters, lmax)
%% initial clusterization

Cmeans = zeros(2,C);
%% calculating change in J
l = 0;
while(l<lmax)
    %% calculating current means
    for i = 1:C
        Cmeans(:,i) = mean(K(:,clusters ==i),2);
    end
    change = false;
    for i = 1:length(K)
        X = K(:,i);
        min_dist = sum((X-Cmeans(:,clusters(i))).^2);

        for j = unique(clusters)
            if(j ~= clusters(i))
                dist = sum((X-Cmeans(:,j)).^2);
                if(dist<min_dist)
                    min_dist = dist;
                    clusters(i) = j;
                    change = true;
                end
            end
        end
    end
    if(change == false)
        break;
    end
    l = l+1;
end

```

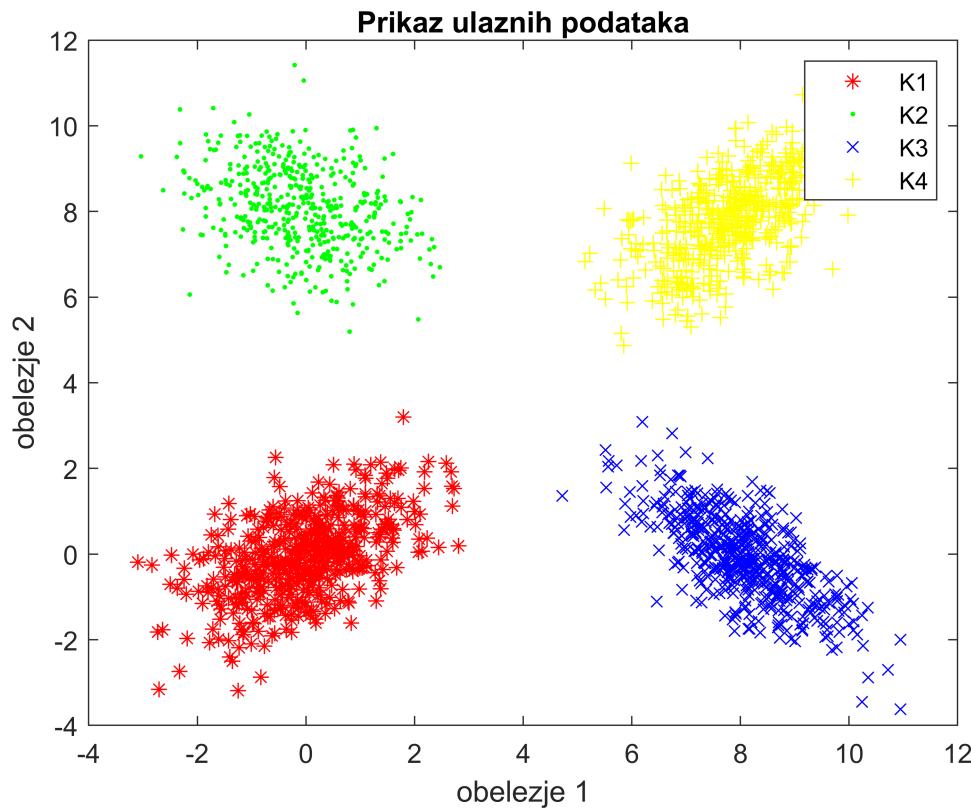
```
final_clustering = clusters;
final_l = 1;
end
```

Generisanje podataka

```
N = 500;

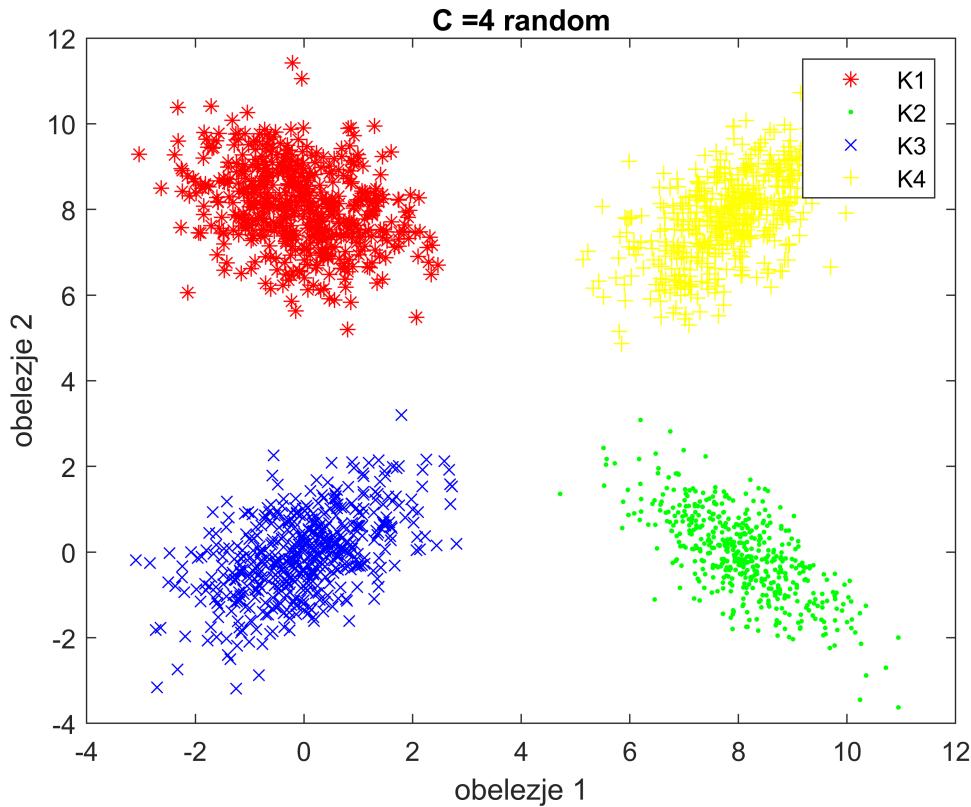
M1 = [0 0]';
S1 = [1 0.5; 0.5 1];
M2 = [0 8]';
S2 = [1 -0.2; -0.2 1];
M3 = [8 0]';
S3 = [1 -0.7; -0.7 1];
M4 = [8 8]';
S4 = [1 0.6; 0.6 1];

K1 = mvnrnd(M1,S1, N);
K2 = mvnrnd(M2,S2, N);
K3 = mvnrnd(M3,S3, N);
K4 = mvnrnd(M4,S4, N);
figure()
plot(K1(:,1),K1(:,2), 'r*')
hold on;
plot(K2(:,1),K2(:,2), 'g.')
hold on;
plot(K3(:,1),K3(:,2), 'bx')
hold on;
plot(K4(:,1), K4(:,2), 'y+')
hold on;
title('Prikaz ulaznih podataka')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')
```



Broj klasa 4, random inicijalizacija, rezultat klasterizacije

```
C = 4;
max_iter = 100;
K = [K1' K2' K3' K4'];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = Cmeans(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C =4 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')
```



Broj klasa 4, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=Cmeans(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_r = iter_mean/20;
disp(iter_mean_4_r)

```

4

Broj klasa 4, inicijalizacija sa 100 dobrih odbiraka, rezultat klasterizacije

```

K_true = [K1(1:100,:)' K2(1:100,:)' K3(1:100,:)' K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:)' K2(101:end,:)' K3(101:end,:)' K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];

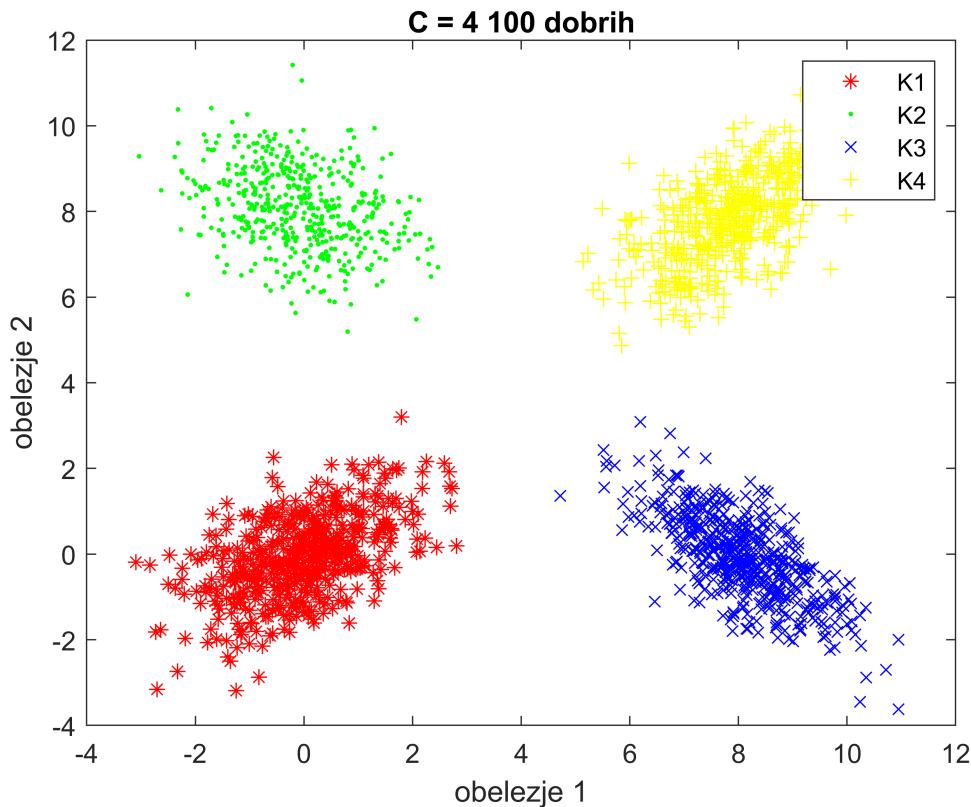
[clusters, iter] = Cmeans(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;

```

```

plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 4 100 dobrih')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')

```



Broj klasa 4, inicijalizacija sa 100 dobrih odbiraka, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=Cmeans(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_100_d = iter_mean/20;
disp(iter_mean_4_100_d)

```

1

Broj klasa 4, inicijalizacija sa 100 losih odbiraka, rezultat klasterizacije

```

K_true = [K3(1:100,:)' K4(1:100,:)' K1(1:100,:)' K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:)' K2(101:end,:)' K3(101:end,:)' K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

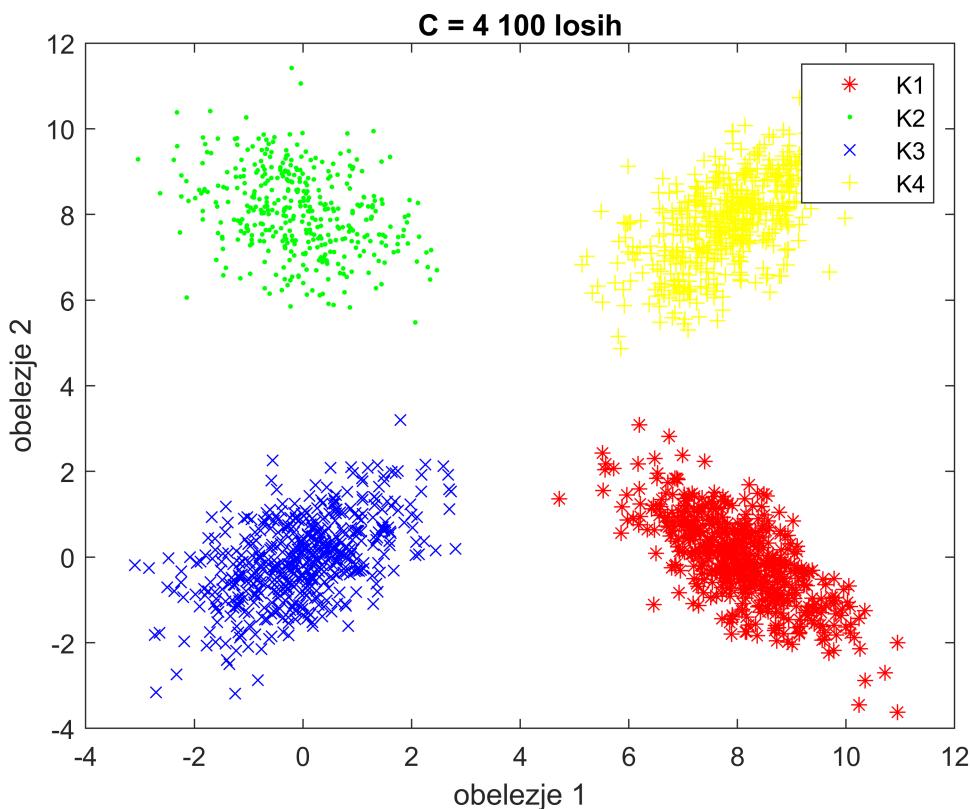
```

```

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];

[clusters, iter] = Cmeans(K, C, initial_clusters, max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 4 100 losih')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')

```



Broj klasa 4, inicijalizacija sa 100 losih odbiraka, srednji broj iteracija

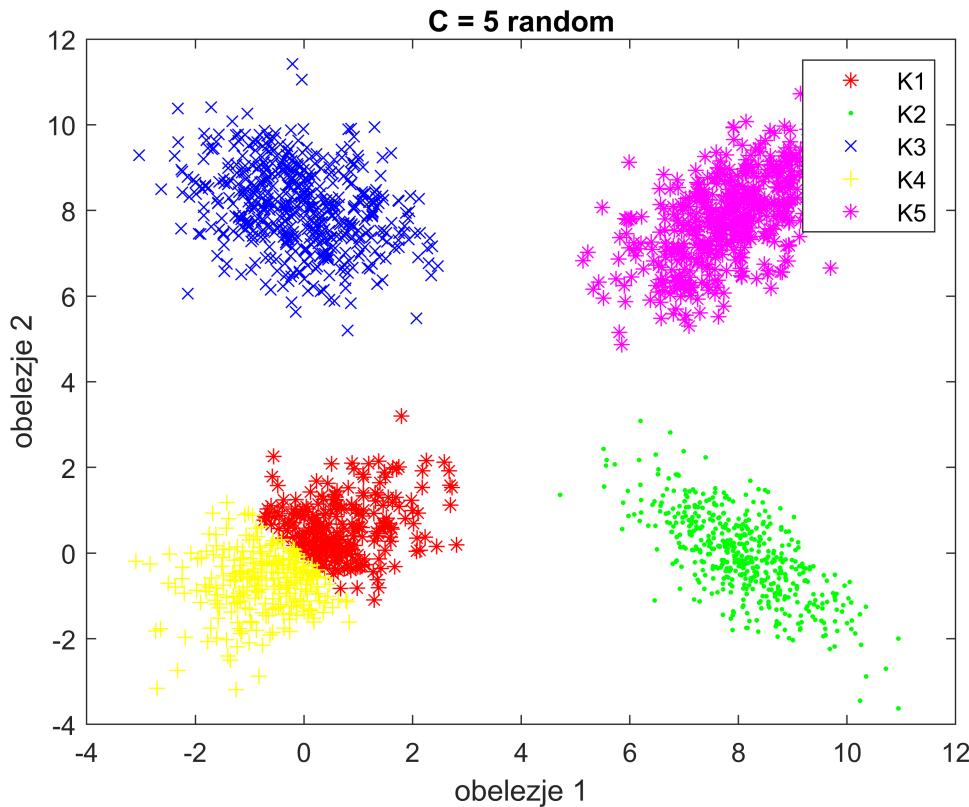
```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=Cmeans(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_100_1 = iter_mean/20;
disp(iter_mean_4_100_1)

```

Broj klasa 5, random inicijalizacija, rezultat klasterizacije

```
C = 5;
max_iter = 100;
K = [K1' K2' K3' K4'];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = Cmeans(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
plot(K(1,clusters == 5),K(2,clusters == 5), 'm*')
hold on;
title('C = 5 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4', 'K5')
```



Broj klasa 5, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=Cmeans(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_5_r = iter_mean/20;
disp(iter_mean_5_r)

```

8

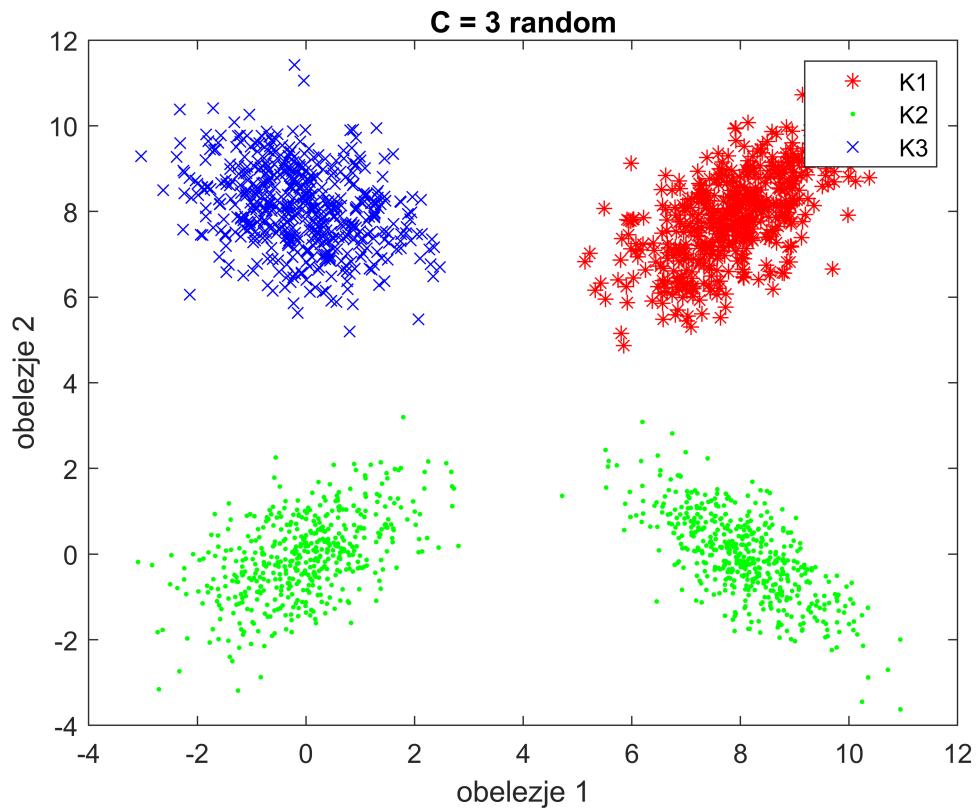
Vidimo da posto je nasem klasifikatoru zadat veci broj klastera od onog koji realno postoji, jedna grupa klastera ce biti prazna ili ce nasilno podeli jednu grupu klastera na dve pravom na sredini klastera.

Broj klasa 3, random inicijalizacija, rezultat klasterizacije

```

C = 3;
max_iter = 100;
K = [K1' K2' K3' K4'];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = Cmeans(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
title('C = 3 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3')

```



Vidimo da s obzirom da u realnosti imamo vise klastera nego sto smo zadali klasifikatoru da nadje on ce spojiti dve grupe u jednu.

Broj klasa 3, random inicializacija, srednji broj iteracija

```
iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=Cmeans(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_3_r = iter_mean/20;
disp(iter_mean_3_r)
```

2

```
A = [iter_mean_3_r,iter_mean_4_r, iter_mean_5_r];
T = array2table(A);
T.Properties.VariableNames(1:3) = {'C=3','C=4','C=5'};
T.Properties.RowNames(1) = {'Zavisnost od broja klasa'};
disp(T)
```

| | C=3 | C=4 | C=5 |
|--------------------------|-----|-----|-----|
| Zavisnost od broja klasa | — | — | — |
| | 2 | 4 | 8 |

```

A = [iter_mean_4_100_d,iter_mean_4_r, iter_mean_4_100_l];
T = array2table(A);
T.Properties.VariableNames(1:3) = {'100 dobrih','random','100 losih'};
T.Properties.RowNames(1) = {'Zavisnost od inicijalizacije'};
disp(T)

```

| | 100 dobrih | random | 100 losih |
|------------------------------|------------|--------|-----------|
| Zavisnost od inicijalizacije | 1 | 4 | 2 |

Mozemo primetiti da zadati broj klasa ne utice na srednji broj iteracija ali incijalizacija itekako utice. Ovo ima smisla jer je nasem klasifikatoru mnogo lakse da klasterizuje podatke ako mu na samom pocetku zadamo 100 dobrih pa mu treba manji broj iteracija, a ako mu zadamo broj losih bice mu mnogo teze da dodje do dobre klasterizacije pa ce mu samim tim biti potreban veci broj iteracija.

MLE klasterizacija

Ovaj metod prepostavlja da su fgv oblika Gausovski raspodeljene pa je samim tim zajednicka fgv svih oblika mesavina gausovskih raspodela. Kriterijumska funkcija je suma zajednickih fgv pri uslovu da je zbir apriornih verovatnca jednak 1.

Izjednacavanjem izvoda po apriornoj verovatnoci, ocekivanju i kovarijacionoj matrici sa nulom dobijaju se formule koriscene u iterativnoj proceduri.

Prvi korak je racunanje ovih parametara a sledeci racunanje aposteriornih verovatnoca na osnovu njih.

Svaki odbirak se dodeljuje onom klasteru cija aposteriorna verovatnoca ima vecu vrednost u njemu.

```

function [clusters, iter] = MLEclustering(K,C, clusters,max_error)
q_prev = zeros(C,length(K));
q = zeros(C,length(K));
M = zeros(2,length(K));
S = zeros(2,2,length(K));
fgv = zeros(C, length(K));
P = zeros(C,1);
for i = 1:C
    M(:,i) = mean(K(:,clusters == i),2);
    S(:,:,:i) = cov(K(:, clusters == i)');
    fgv(i,:) = 1/(2*pi)/sqrt(det(S(:,:,:i)))*...
        exp(-0.5.*diag(((K-M(:,i))'*inv(S(:,:,:i))*(K-M(:,i))))');
    P(i) = 1/C;
end
iter = 0;
while(1)
    for i = 1:C
        q(i,:) = P(i)*fgv(i,:)./sum(fgv.*P,1);

        P(i) = mean(q(i,:),2);
        M(:,i) = mean(q(i,:).*K,2)/P(i);
        for j = 1:length(K)
            S(:,:,:i) = S(:,:,:i)+q(i,j)*((K(:,j)-M(:,i))*(K(:,j)-M(:,i))')/length(K);
        end
        S(:,:,:i) = S(:,:,:i)/length(K);
        fgv(i,:) = 1/(2*pi)/sqrt(det(S(:,:,:i)))*...
            exp(-0.5.*diag((K-M(:,i))'*inv(S(:,:,:i))*(K-M(:,i))))';
        q(i,:) = fgv(i,:)./sum(fgv,1);
    end
    if(max(abs(q_prev - q))<=max_error)
        break;
    end
    q_prev = q;
    iter = iter+1;
    for j = 1:length(K)
        [nebitno,clusters(j)] = max(q(:,j));
    end
end

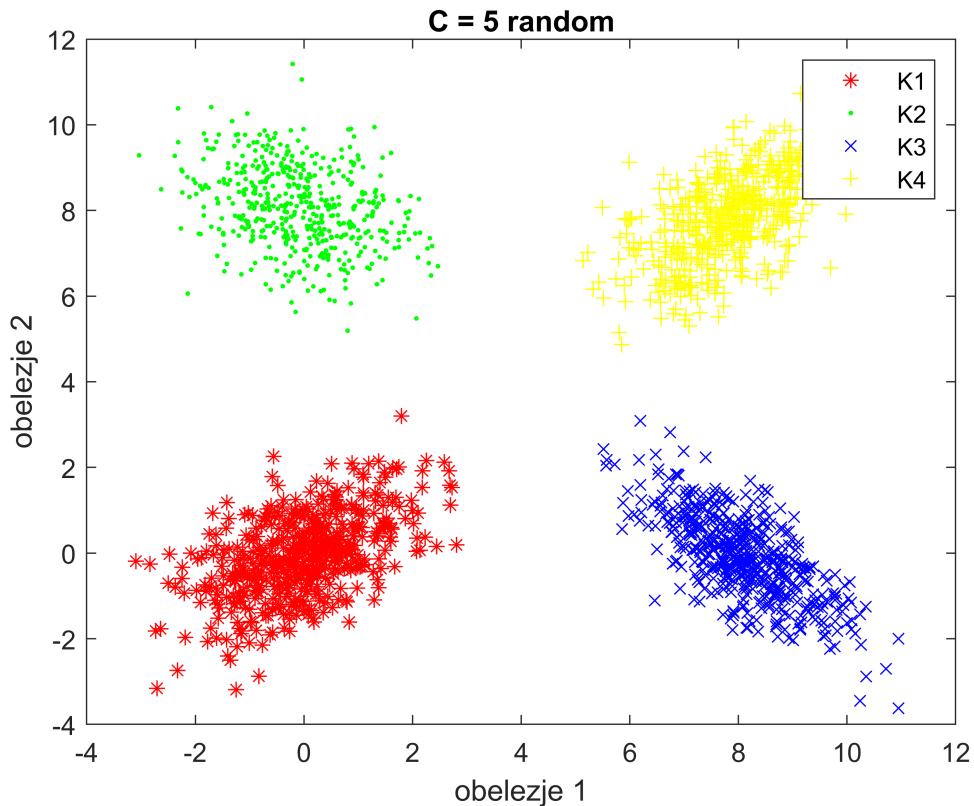
```

Broj klasa 5, random inicijalizacija, rezultat klasterizacije

```
C = 5;
K_true = [K1(1:100,:)'; K2(1:100,:)'; K3(1:100,:)'; K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:)'; K2(101:end,:)'; K3(101:end,:)'; K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];
```

```
max_error = 1e-2;
[clusters,iter] = MLEclustering(K,C,initial_clusters, 1e-2);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 5 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')
```



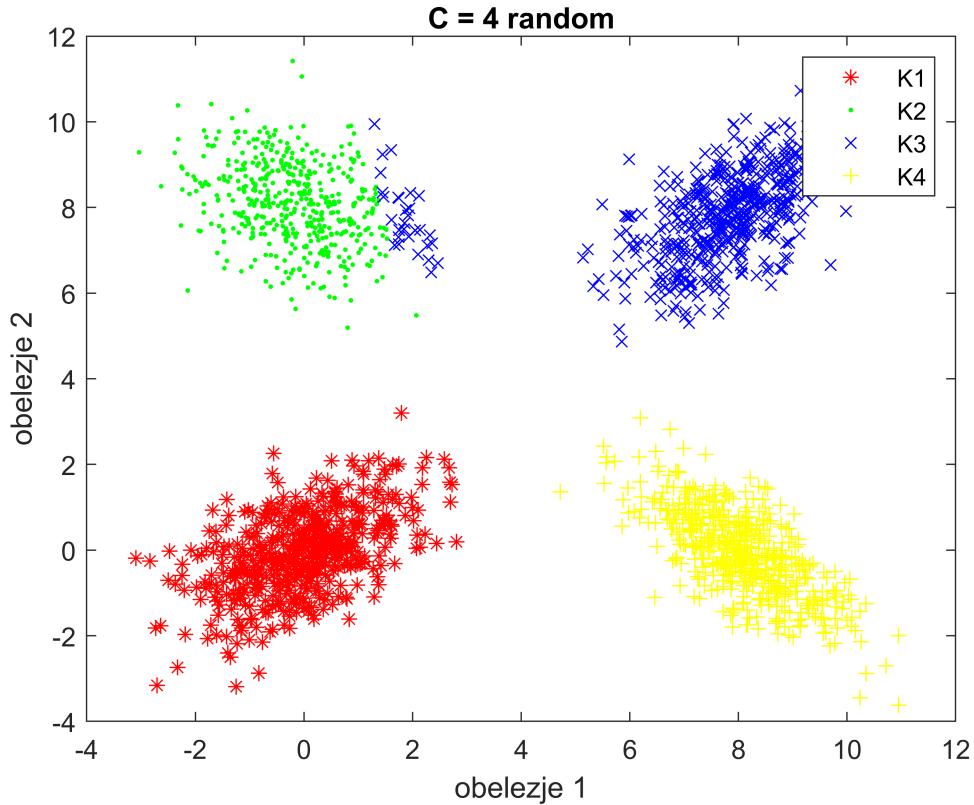
Broj klasa 5, random inicijalizacija, srednji broj iteracija

```
iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=MLEclustering(K,C,initial_clusters, max_error);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_5_r = iter_mean/20;
disp(iter_mean_5_r)
```

32

Broj klasa 4, random inicijalizacija, rezultat klasterizacije

```
C = 4;
max_iter = 100;
K = [K1' K2' K3' K4'];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = MLEclustering(K,C,initial_clusters, 1e-2);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 4 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')
```



Broj klasa 4, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=MLEclustering(K,C,initial_clusters, 1e-2);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_r = iter_mean/20;
disp(iter_mean_4_r)

```

1

Broj klasa 3, random inicijalizacija, rezultat klasterizacije

```

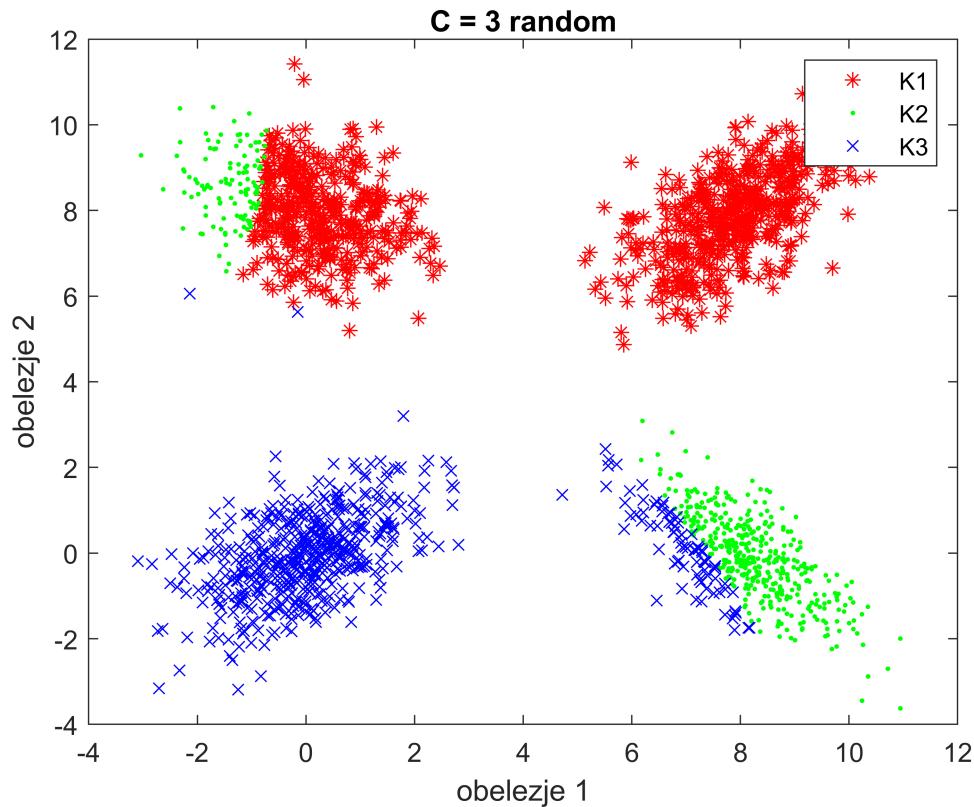
C = 3;
K_true = [K1(1:100,:)'; K2(1:100,:)'; K3(1:100,:)'; K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:)'; K2(101:end,:)'; K3(101:end,:)'; K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));
%%
max_iter = 100;
K = [K1' K2' K3' K4'];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = MLEclustering(K,C,initial_clusters, 1e-2);
figure()

```

```

plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
title('C = 3 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3')

```



Broj klasa 3, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=MLEclustering(K,C,initial_clusters, 1e-2);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_3_r = iter_mean/20;
disp(iter_mean_3_r)

```

2

Broj klasa 4, inicijalizacija sa 100 dobrih odbiraka, rezultat klasterizacije

C = 4

C = 4

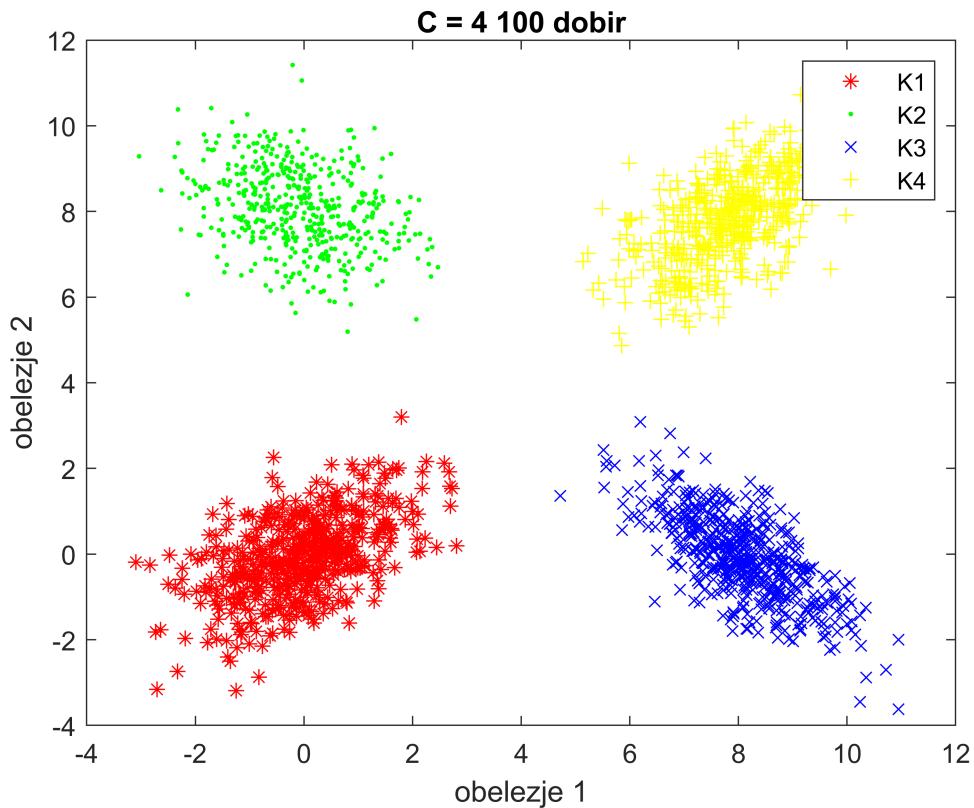
```

K_true = [K1(1:100,:)'; K2(1:100,:)'; K3(1:100,:)'; K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:)', K2(101:end,:)', K3(101:end,:)', K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C, 1, length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];

[clusters, iter] = MLEclustering(K,C,initial_clusters, 1e-2);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 4 100 dobir')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')

```



Broj klasa 4, inicijalizacija sa 100 dobrih odbiraka, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=MLEclustering(K,C,initial_clusters, 1e-2);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_100_d = iter_mean/20;
disp(iter_mean_4_100_d)

```

16

Broj klasa 4, inicijalizacija sa 100 losih odbiraka, rezultat klasterizacije

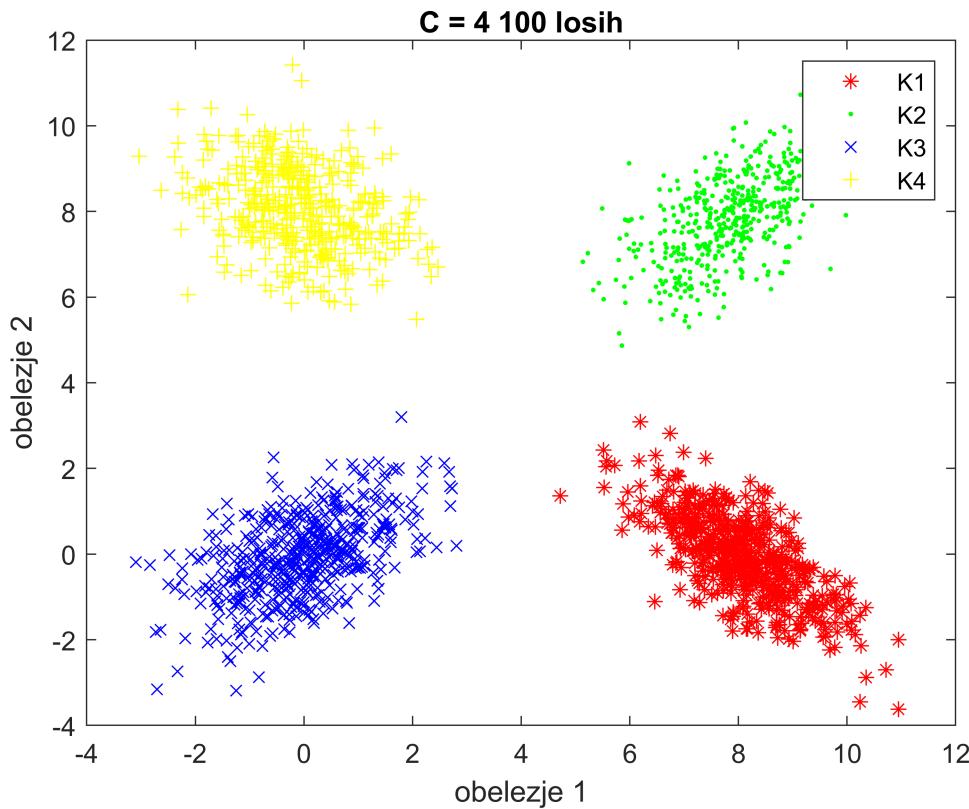
```

K_true = [K3(1:100,:) K4(1:100,:) K1(1:100,:) K4(1:100,:)'];
clusters_true = [ones(1,100),2*ones(1,100),3*ones(1,100),4*ones(1,100)];
K_rand = [K1(101:end,:) K2(101:end,:) K3(101:end,:) K4(101:end,:)'];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];

[clusters, iter] = MLEclustering(K,C,initial_clusters, 1e-2);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
title('C = 4 100 losih')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4')

```



Broj klasa 4, inicijalizacija sa 100 losih odbiraka, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=MLEclustering(K,C,initial_clusters, 1e-2);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_4_100_l = iter_mean/20;
disp(iter_mean_4_100_l)

```

19

```

A = [iter_mean_3_r,iter_mean_4_r, iter_mean_5_r];
T = array2table(A);
T.Properties.VariableNames(1:3) = {'C=3','C=4','C=5'};
T.Properties.RowNames(1) = {'Zavisnost od broja klasa'};
disp(T)

```

| Zavisnost od broja klasa | C=3 | C=4 | C=5 |
|--------------------------|-----|-----|-----|
| | — | — | — |
| Zavisnost od broja klasa | 2 | 1 | 32 |

```

A = [iter_mean_4_100_d,iter_mean_4_r, iter_mean_4_100_l];
T = array2table(A);
T.Properties.VariableNames(1:3) = {'100 dobrih','random','100 losih'};

```

```
T.Properties.RowNames(1) = {'Zavisnost od inicijalizacije'};  
disp(T)
```

| | 100 dobrih | random | 100 losih |
|------------------------------|------------|--------|-----------|
| Zavisnost od inicijalizacije | 16 | 1 | 19 |

Prvo, primetimo da je MLE znatno osjetljiviji na zadati broj klasa. Ako mu zadamo pogresan broj klasa klasterizacija nece biti mnogo smislena(cak nece jedan klaster podeliti na dva ili spojiti dva klastera) i treba mu mnogo vise iteracija da bi je izvrsio. Takodje, najbolje rezultate dobijamo za random inicijalizaciju.

Metod normalne dekompozicije

Metod kvadratne dekompozicije je jedan od metoda normalne dekompozicije. On je jako slican C-means metodu jedina razlika je sto kriterijumska funkcija vise nije rastojanje od srednje vrednosti vec aposteriorna verovatnoca u slucaju Gausove raspodele.

- Zahteva apriorno znanje, ako ga nemamo, rezultat klasterizacije moze biti jako los
- zahteva da znamo broj klastera
- Moguce je da klasifikaciona linija bude nelinearna kriva za razliku od C means gde je to bila prava ALI moze biti samo kriva drugog reda
- Nije zagarantovan globalni minimum
- Slozeniji je od K-mean i zahteva veci broj iteracija

```

function [final_clustering, final_l] = NormalDecomposition(K,C,clusters, lmax)
    %% initial clusterization
    P = zeros(1,C);
    M = zeros(2,C);
    S = zeros(2,2,C);
    %% calculating change in J
    l = 0;
    while(l<lmax)
        %% calculating current means
        for i = 1:C
            P(i) = sum(clusters == i)/length(clusters);
            M(:,i) = mean(K(:, clusters == i),2);
            S(:,:,i) = cov(K(:, clusters ==i)');
        end
        change = false;
        for i = 1:length(K)
            X = K(:,i);
            id = clusters(i);
            min_dist = -log(P(id))+0.5*log(det(S(:,:,id)))+0.5*(X-M(:,id))'*inv(S(:,:,id));
            for j = unique(clusters)
                if(j ~= id)
                    dist = -log(P(j))+0.5*log(det(S(:,:,j)))+0.5*(X-M(:,j))'*inv(S(:,:,j));
                    if(dist<min_dist)
                        min_dist = dist;
                        clusters(i) = j;
                        change = true;
                    end
                end
            end
        end
    end
end

```

```

        end
    end
end
if(change == false)
    break;
end
l = l+1;
end

final_clustering = clusters;
final_l = l;
end

```

Generisanje nelinearno separabilnih podataka

```

N = 500;
x = rand(1,500)*10^-5;
rho = rand(1,500);
theta = rand(1,500)*pi/3;

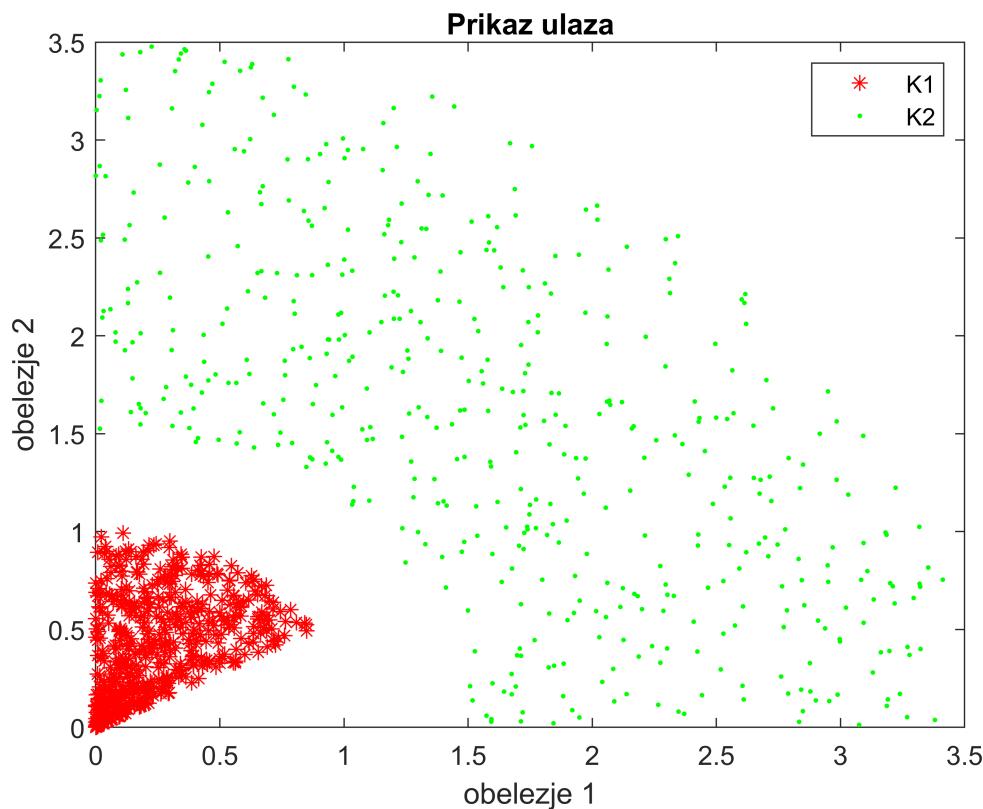
K1 = [rho.*sin(theta);rho.*cos(theta)];

rho = rand(1,500)*2+1.5;
theta = rand(1,500)*pi/2;

K2 = [rho.*sin(theta);rho.*cos(theta)];

figure()
plot(K1(1,:),K1(2,:),'r*')
hold on;
plot(K2(1,:),K2(2,:),'g.')
hold on;
title('Prikaz ulaza')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2')

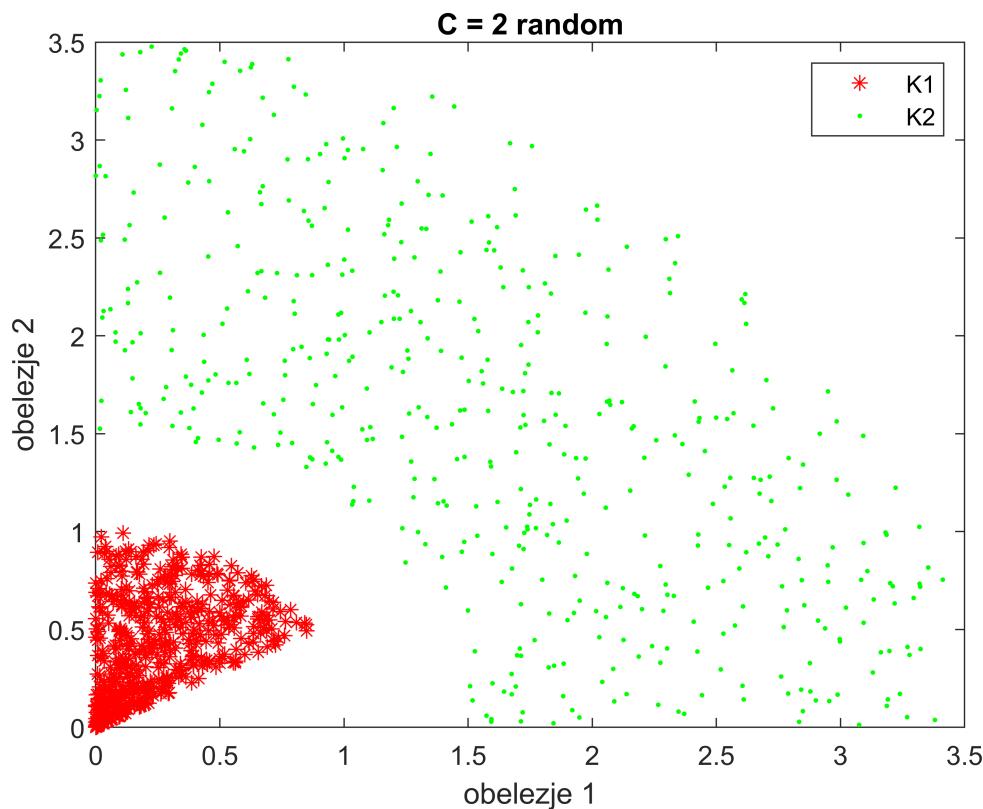
```



Broj klasa 2, random inicijalizacija, rezultat klasterizacije

```
C = 2;
max_iter = 100;
K = [K1 K2];
K = K(:, randperm(size(K, 2)));

initial_clusters = randi(C,1,length(K));
[clusters, iter] = NormalDecomposition(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
title('C = 2 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2')
```



Broj klasa 2, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=NormalDecomposition(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_2_r = iter_mean/20;
disp(iter_mean_2_r)

```

7

S obzirom da ovaj metod podrazumeva normalnu raspodelu pokusace da podeli skup podataka na dva dela kao da su oni Gausovski raspodeljeni zbog cega ne uspeva dobro da klasterize nase podatke.

Broj klasa 2,inicijalizacija sa 100 dobrih, rezultat klasterizacije

```

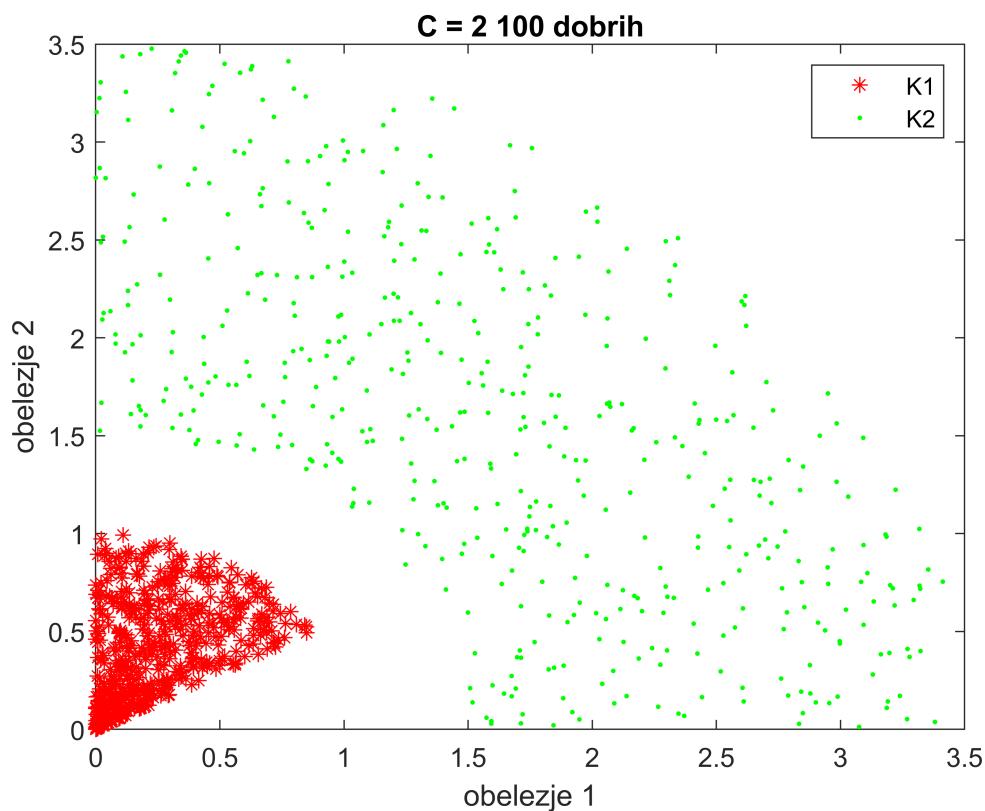
K_true = [K1(:,1:100) K2(:,1:100)];
clusters_true = [ones(1,100),2*ones(1,100)];
K_rand = [K1(:,101:end) K2(:,101:end)];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];

```

Broj klasa 2, inicijalizacija sa 100 dobrih, srednji broj iteracija

```
[clusters, iter] = NormalDecomposition(K, C, initial_clusters, max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
title('C = 2 100 dobrih')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2')
```



```
iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=NormalDecomposition(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_2_100_d = iter_mean/20;
disp(iter_mean_2_100_d)
```

3

Kada mu zadamo sto dobrih za pocetne uslove klasterizacija je dobro izvrsena, ali 100 semplova na 500 je 1/5 sto u realnosti ako nam je potrebna klasterizacija a ne klasifikacija necemo moci da imamo.

Broj klasa 2, inicializacija sa 100 losih, rezultat klasterizacije

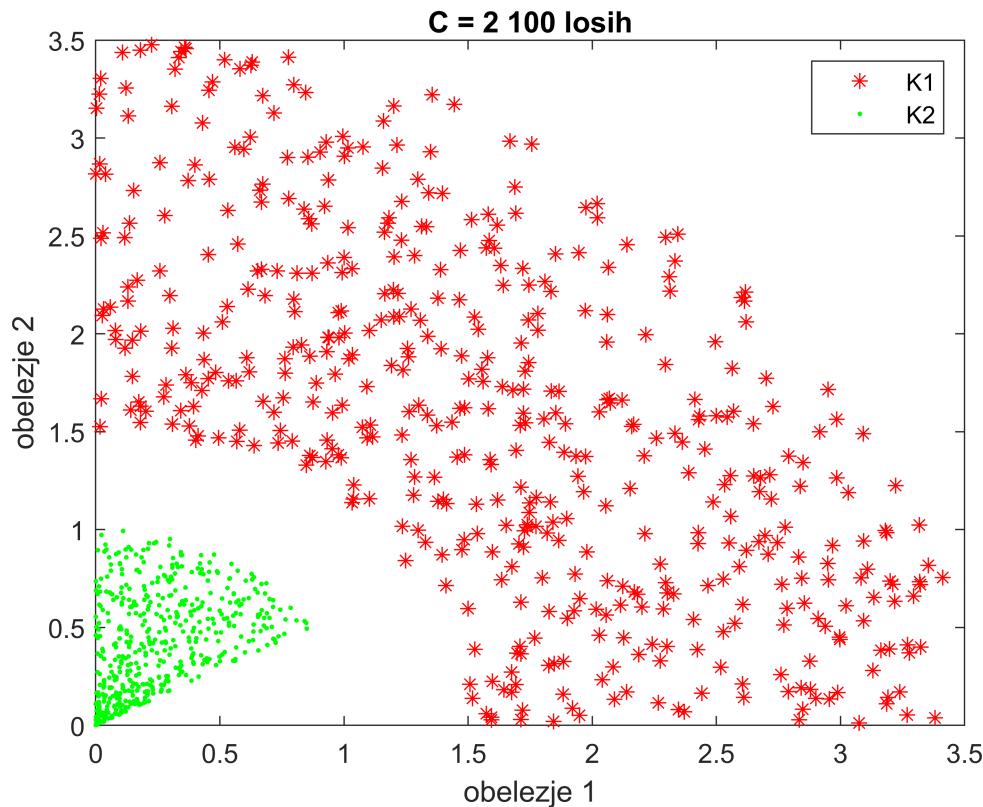
```
K_true = [K2(:,1:100) K1(:,1:100) ];
```

```

clusters_true = [ones(1,100),2*ones(1,100)];
K_rand = [K1(:,101:end) K2(:,101:end)];
K_rand = K_rand(:, randperm(size(K_rand, 2)));
clusters_rand = randi(C,1,length(K_rand));

K = [K_true, K_rand];
initial_clusters = [clusters_true, clusters_rand];
[clusters, iter] = NormalDecomposition(K, C, initial_clusters, max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
title('C = 2 100 losih')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2')

```



Broj klasa 2, inicijalizacija sa 100 losih, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=NormalDecomposition(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_2_100_1 = iter_mean/20;
disp(iter_mean_2_100_1)

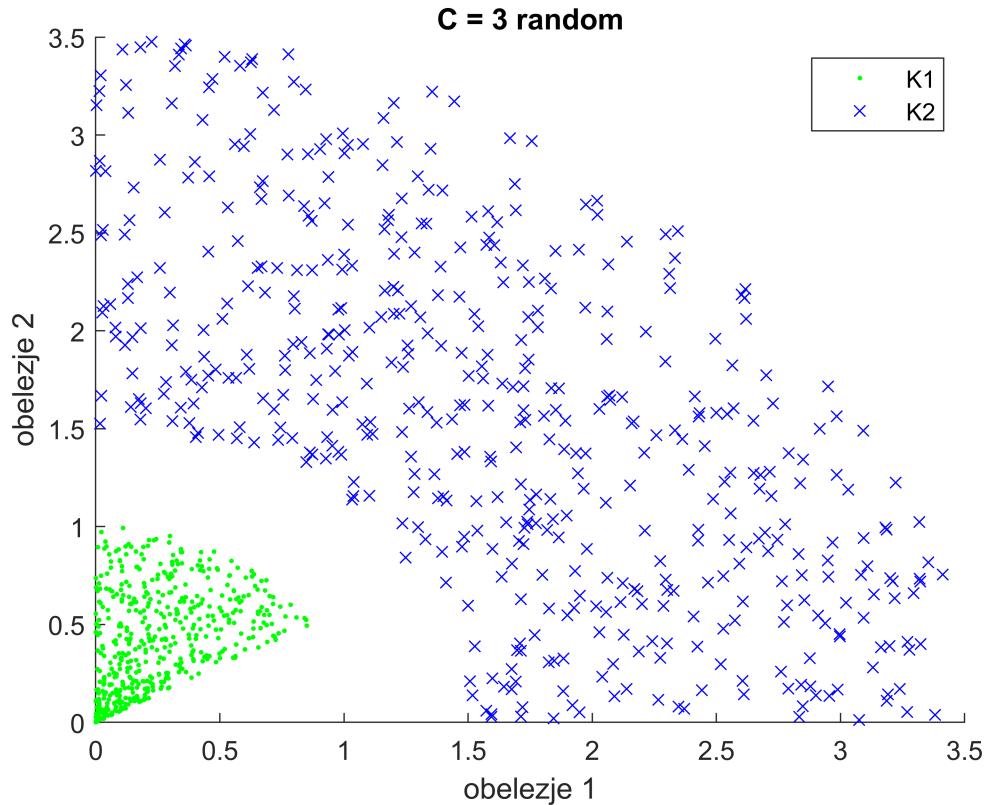
```

Za slučaj dva klastera, broj losih odbiraka dace isti rezultat kao ako mu damo dobre zato sto podatke treba da podelimo samo na dva skupa

Broj klasa 3, random inicijalizacija, rezultat klasterizacije

```
C = 3;
max_iter = 100;
K = [K1 K2];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = NormalDecomposition(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
title('C = 3 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3')
```

Warning: Ignoring extra legend entries.



Broj klasa 3, random inicijalizacija, srednji broj iteracija

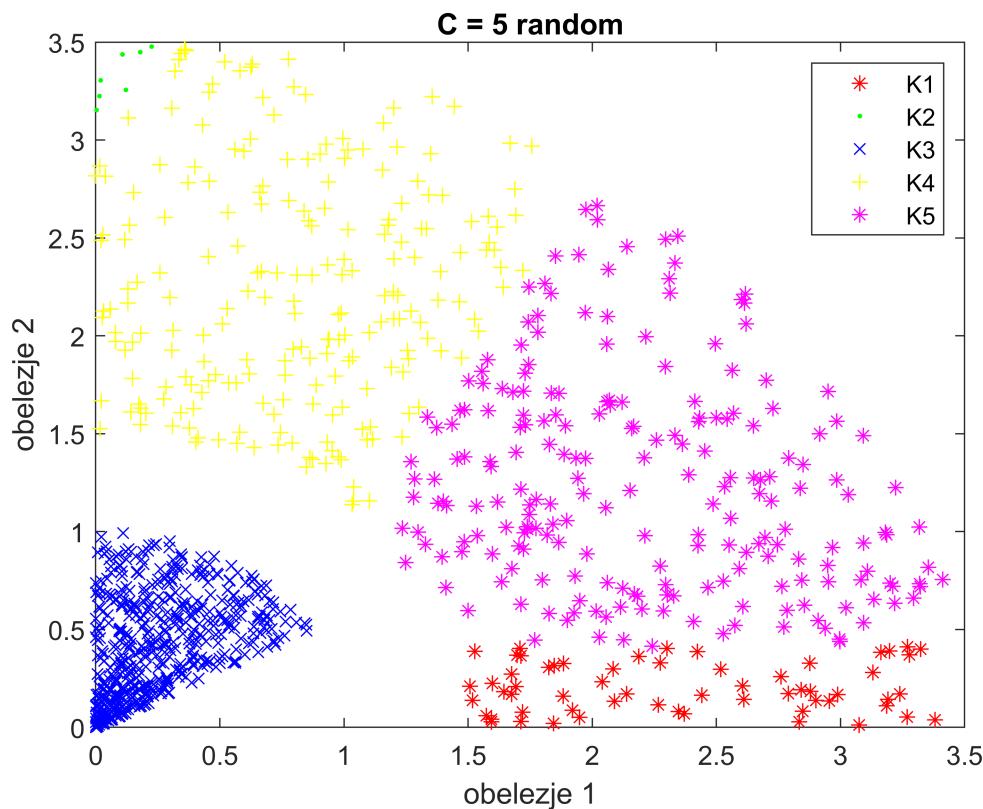
```
iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=NormalDecomposition(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_3_r = iter_mean/20;
disp(iter_mean_3_r)
```

14

Primetimo da se sve vreme klaster podataka oko centra mnogo lakse klasterizuje jer ga mozemo obuhvatiti gausovom raspodelom dok prsten ne mozemo jer ima supljinu u centru koju Gausova raspodela mora da obezbedi. Klasterizacija na tri dela je mnogo laksa jer mozemo postaviti dva centra na prstenu pa obuhvatiti Gausovom rapsodelom svaki od njih. Ipak taj nacin klasterizacije nije intuitivan jer covek ne bi percipirao te dve vrste podataka kao da pripadaju razlicitom skupu.

Broj klasa 5, random inicijalizacija, rezultat klasterizacije

```
C = 5;
max_iter = 100;
K = [K1 K2];
K = K(:, randperm(size(K, 2)));
initial_clusters = randi(C,1,length(K));
[clusters, iter] = NormalDecomposition(K, C,initial_clusters,max_iter);
figure()
plot(K(1,clusters == 1),K(2,clusters == 1), 'r*')
hold on;
plot(K(1,clusters == 2),K(2,clusters == 2), 'g.')
hold on;
plot(K(1,clusters == 3),K(2,clusters == 3), 'bx')
hold on;
plot(K(1,clusters == 4),K(2,clusters == 4), 'y+')
hold on;
plot(K(1,clusters == 5),K(2,clusters == 5), 'm*')
hold on;
title('C = 5 random')
xlabel('obelezje 1')
ylabel('obelezje 2')
legend('K1', 'K2', 'K3', 'K4', 'K5')
```



Broj klasa 5, random inicijalizacija, srednji broj iteracija

```

iter_mean = 0;
for i = 1:20
    [nebitno, curr_iter]=NormalDecomposition(K,C,initial_clusters, max_iter);
    iter_mean = iter_mean+curr_iter;
end
iter_mean_5_r= iter_mean/20;
disp(iter_mean_5_r)

```

12

Za slučaj 5 klasa vazi isto kao za 3.

Sumarni prikaz srednjeg broja iteracija

```

A = [iter_mean_2_r,iter_mean_3_r, iter_mean_5_r];
T = array2table(A);
T.Properties.VariableNames(1:3) = {'C=2','C=3','C=5'};
T.Properties.RowNames(1) = {'Zavisnost od broja klasa'};
disp(T)

```

| | C=2 | C=3 | C=5 |
|--------------------------|-----|-----|-----|
| Zavisnost od broja klasa | 7 | 14 | 12 |

```
A = [iter_mean_2_100_d,iter_mean_2_r, iter_mean_2_100_1];
```

```
T = array2table(A);
T.Properties.VariableNames(1:3) = {'100 dobrih','random','100 losih'};
T.Properties.RowNames(1) = {'Zavisnost od inicializacije'};
disp(T)
```

| | 100 dobrih | random | 100 losih |
|-----------------------------|------------|--------|-----------|
| Zavisnost od inicializacije | 3 | 7 | 2 |

Srednji broj iteracija raste sa brojem klasa, a najteza je random inicializacija i potreban je veci broj iteracija nego kod C-means.