

UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

MAŠINSKO UČENJE (MACHINE LEARNING)

Mašinsko učenje (engl. *Machine Learning*, *ML*) je grana vještice inteligencije koja omogućava računarima da uče iz podataka i poboljšavaju svoje performanse bez eksplicitnog programiranja.

Osnovna ideja je da algoritam pronađe šablove (pattern-e) i odnose među varijablama kako bi mogao donositi odluke, predviđanje ili klasifikacije.

Mašinsko učenje se dijeli na tri glavne kategorije:

a) Nadzirano učenje (engl. Supervised Learning)

U ovom pristupu algoritam uči na označenim podacima (engl. *labeled data*), gdje su poznati ulazi i odgovarajući izlazi.

- **Cilj:** Naučiti funkciju $f(x) \approx y$ koja preslikava ulazne podatke x u željene izlaze y .
- **Primjene:** Predikcija napona, temperature, cijene, klasifikacija kvarova, itd.
- **Algoritmi:** Linearna regresija, logistička regresija, SVM, k-NN, neuronske mreže.

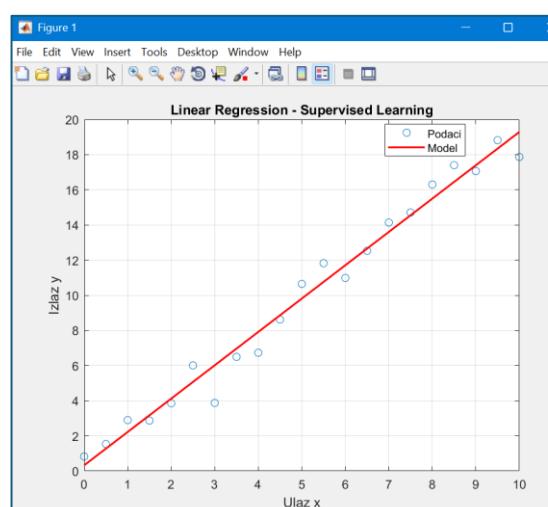
Primjer1

```
% Nadzirano učenje - Linearna regresija
x = (0:0.5:10)'; % Ulazni podaci
y = 2*x + randn(size(x)); % Izlazni podaci
sa šumom

% Treniranje modela
mdl = fitlm(x, y);

% Predikcija i prikaz
x_pred = (0:0.1:10)';
y_pred = predict(mdl, x_pred);

plot(x, y, 'o');
hold on;
plot(x_pred, y_pred, 'r', 'LineWidth', 1.5);
title('Linearna regresija - Nadzirano učenje');
xlabel('Ulaz x'); ylabel('Izlaz y');
legend('Podaci', 'Model');
grid on;
```



Slika 1

Dobijeni graf prikazuje tačkaste podatke i crvenu liniju koja predstavlja naučeni model linearne regresije.

Primjer2

Primjer nadziranog učenja za PV sistem, gdje su ulazi sunčeva radijacija (G) i temperatura (T), a izlazi napon (V) i struja (I) PV panela. Simulira se stvarni hibridni energetski sistem i pokazuje kako neuronska mreža može predviđati performanse panela.

Generisanje sintetičkih podataka

- Ulazi:
G – sunčeva radijacija [W/m^2] (100–1000 W/m^2)
T – temperatura [$^\circ\text{C}$] (15–40 $^\circ\text{C}$)
- Izlazi:
V – napon PV panela [V]
I – struja PV panela [A]

Koristimo linearno-nelinearni model PV panela:

$$V = V_{oc} - k_v(T - 25) - \alpha I, I = I_{sc} \cdot \frac{G}{1000}$$

```
clc; clear; close all;

%% 1. Generisanje ulaza (G i T)
num_samples = 200;
G = 100 + (1000-100).*rand(1,num_samples); % Sunčeva radijacija [W/m?]
T = 15 + (40-15).*rand(1,num_samples); % Temperatura [°C]

inputs = [G; T]; % Dimenzija: [2 × num_samples]

%% 2. Generisanje izlaza (V i I) sa šumom
V_oc = 38; % Otvoreni napon PV panela [V]
I_sc = 5; % Struja kratkog spoja [A]
k_v = 0.3; % Koeficijent temperature
alpha = 0.01; % Interni otpor

I = I_sc .* (G./1000) + 0.05*randn(1,num_samples); % Sintetički I sa šumom
V = V_oc - k_v*(T-25) - alpha*I + 0.1*randn(1,num_samples); % Sintetički V sa šumom

outputs = [V; I]; % Dimenzija: [2 × num_samples]

%% 3. Podjela na trening i test
train_ratio = 0.7;
idx = 1:round(num_samples*train_ratio);

inputs_train = inputs(:,idx);
outputs_train = outputs(:,idx);

inputs_test = inputs(:,idx(end)+1:end);
outputs_test = outputs(:,idx(end)+1:end);

%% 4. Kreiranje neuronske mreže
hidden_neurons = 10;
net = fitnet(hidden_neurons); % Mreža za regresiju

net.trainParam.showWindow = true;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-4;

%% 5. Treniranje mreže
[net, tr] = train(net, inputs_train, outputs_train);

%% 6. Testiranje
```

```

outputs_pred = net(inputs_test);

%% 7. Vizuelizacija
figure;

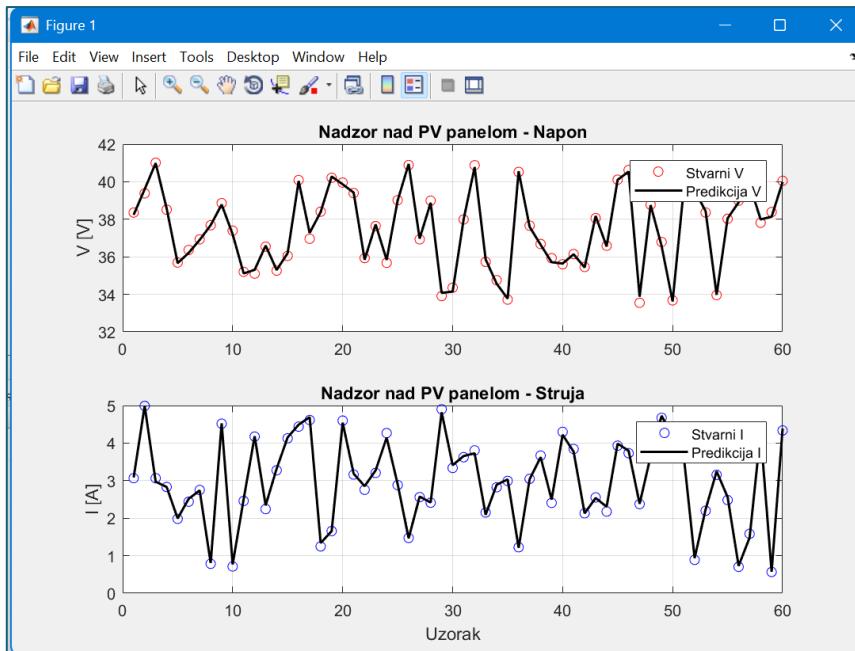
subplot(2,1,1);
plot(outputs_test(1,:), 'ro', 'DisplayName', 'Stvarni V');
hold on;
plot(outputs_pred(1,:), 'k-', 'LineWidth', 1.5, 'DisplayName', 'Predikcija V');
ylabel('V [V]');
title('Nadzor nad PV panelom - Napon');
legend; grid on;

subplot(2,1,2);
plot(outputs_test(2,:), 'bo', 'DisplayName', 'Stvarni I');
hold on;
plot(outputs_pred(2,:), 'k-', 'LineWidth', 1.5, 'DisplayName', 'Predikcija I');
ylabel('I [A]');
xlabel('Uzorak');
title('Nadzor nad PV panelom - Struja');
legend; grid on;

%% 8. Izračun MSE
mse_V = mse(outputs_test(1,:)-outputs_pred(1,:));
mse_I = mse(outputs_test(2,:)-outputs_pred(2,:));
fprintf('MSE V = %.4f, MSE I = %.4f\n', mse_V, mse_I);

```

MSE V = 0.0157, MSE I = 0.0033



Slika 2

Primjer3

Nadzor i predikcija performansi vjetroturbine pomoću neuronske mreže. Snaga vjetra se opisuje relacijom:

$$P = \frac{1}{2} \rho A C_p v^3$$

gdje su:

- $\rho = 1.225 \text{ kg/m}^3$ – gustina zraka,
- $A = \pi r^2$ – površina rotora,
- C_p – koeficijent snage (0.25–0.45),
- v – brzina vjetra [m/s].

Moment generatora se određuje kao: $T = \frac{P}{\omega}$, gdje je ω ugaona brzina, proporcionalna brzini vjetra.

```
clc; clear; close all;

%% 1. Generisanje ulaznih podataka
num_samples = 200;
v = 2 + (15-2).*rand(1,num_samples); % Brzina vjetra [m/s]

%% 2. Parametri turbine
rho = 1.225; % Gustina zraka [kg/m^3]
r = 40; % Poluprečnik rotora [m]
A = pi * r^2; % Površina rotora
Cp = 0.4; % Koeficijent snage
k_w = 0.12; % Koeficijent proporcionalnosti za ugaonu brzinu

%% 3. Izračunavanje izlaza (P i T)
P = 0.5 * rho * A * Cp .* (v.^3) / 1e6; % Snaga u MW
omega = k_w * v; % Ugaona brzina [rad/s]
T = (P * 1e6) ./ omega; % Moment [Nm]

% Dodavanje šuma radi realnosti
P = P + 0.05*randn(1,num_samples);
T = T + 10*randn(1,num_samples);

inputs = v;
outputs = [P; T]; % Izlazi: snaga i moment

%% 4. Podjela na trening i test
train_ratio = 0.7;
idx = 1:round(num_samples*train_ratio);
x_train = inputs(:,idx);
y_train = outputs(:,idx);
x_test = inputs(:,idx(end)+1:end);
y_test = outputs(:,idx(end)+1:end);

%% 5. Kreiranje i treniranje neuronske mreže
net = fitnet(10); % 10 neurona u skrivenom sloju
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-4;

[net, tr] = train(net, x_train, y_train);

%% 6. Testiranje i poređenje
y_pred = net(x_test);

figure;
subplot(2,1,1);
plot(y_test(1,:), 'ro', 'DisplayName', 'Stvarna P');
hold on;
plot(y_pred(1,:), 'k-', 'LineWidth', 1.3, 'DisplayName', 'Predikcija P');
```

```

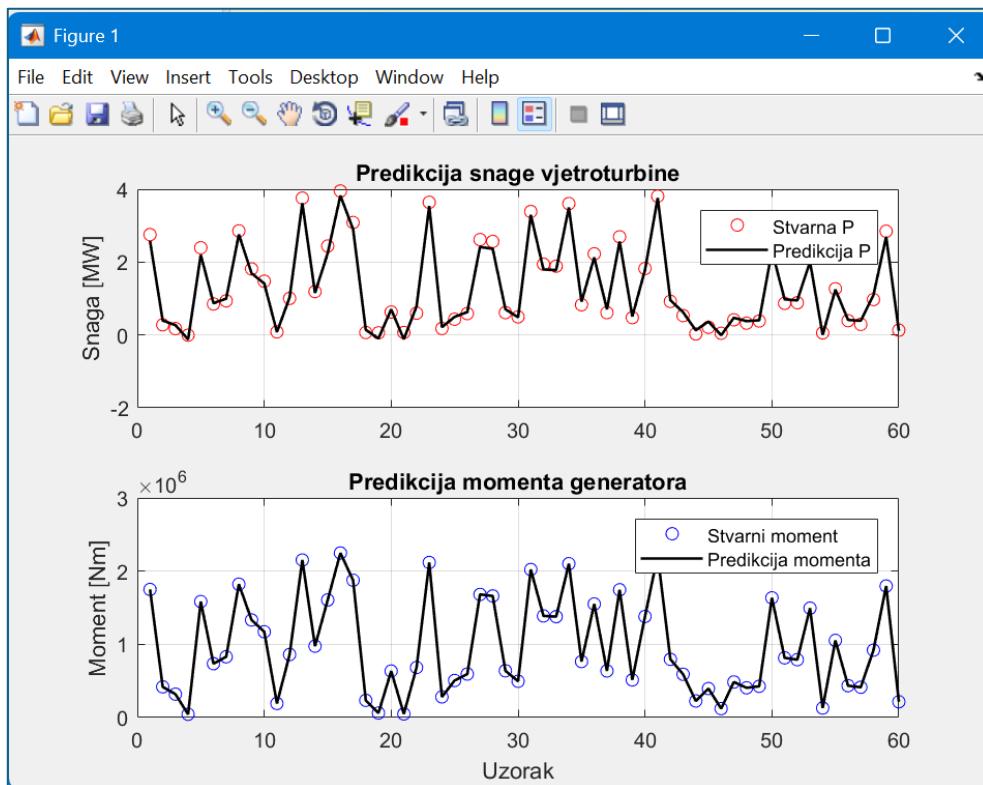
ylabel('Snaga [MW]');
title('Predikcija snage vjetroturbine');
legend; grid on;

subplot(2,1,2);
plot(y_test(2,:), 'bo', 'DisplayName', 'Stvarni moment');
hold on;
plot(y_pred(2,:), 'k-', 'LineWidth', 1.3, 'DisplayName', 'Predikcija
momenta');
ylabel('Moment [Nm]');
xlabel('Uzorak');
title('Predikcija momenta generatora');
legend; grid on;

%% 7. Greška
mse_P = mse(y_test(1,:) - y_pred(1,:));
mse_T = mse(y_test(2,:) - y_pred(2,:));
fprintf('MSE snaga = %.4f, MSE moment = %.4f\n', mse_P, mse_T);

```

MSE snaga = 0.0131, MSE moment = 133.0264



Slika 3

Primjer4 Statička regresija: predikcija temperature prostorije

Potrebno je kreirati model koji može procjeniti temperaturu prostorije Troom, u datom trenutku, na osnovu vrijednosti senzora koji mijere:

- Tout — vanjska temperatura (°C)
- HVAC — trenutno opterećenje/izlaz HVAC sistema (0-1 ili 0-100% snage)
- Occ — broj prisutnih osoba
- RH — relativna vlažnost (%)

Model treba da uči na osnovu historijskih podataka, a nakon treniranja da se može koristiti za predikciju temperature u stvarnom vremenu, što omogućava inteligentno upravljanje HVAC sistemom radi poboljšanja energetske efikasnosti i udobnosti korisnika.

HVAC(engl. Heating, Ventilation, and Air Conditioning - Grijanje, ventilacija i klimatizacija) integrисани sistem koji se koristi za upravljanje temperaturom, vlažnošću i kvalitetom zraka u zatvorenim prostorima (zgradama, laboratorijama, učionicama, server sobama itd.)

```
clc; clear; close all;

% 1) Generisanje sintetičkih podataka (realistične varijacije)
N = 500;
t = (1:N);

% Vanjska temp: dnevni ciklus (sinusoidno) + promjene
Tout = 10 + 8*sin(2*pi*t/144) + 0.8*randn(1,N); % °C

% HVAC output zavisi od kontrole (nekontrolisano sintetički)
HVAC = 0.5 + 0.4*sin(2*pi*(t+20)/200) + 0.1*randn(1,N);
HVAC = min(max(HVAC, 0), 1);

% Broj osoba (diskretan) slučajan raspored
Occ = poissrnd(1.2,1,N);
Occ(Occ>5)=5;

% Relativna vlažnost (RH)
RH = 40 + 20*sin(2*pi*(t+50)/300) + 3*randn(1,N);
RH = min(max(RH,20),90);

% Stvarna sobna temp model
alpha0 = 5.0;
alpha1 = 0.55; % utjecaj vanjske temperature
alpha2 = -6.0; % HVAC hlađi (negativan utjecaj ako HVAC=1 -> niža temp)
alpha3 = 0.2; % svaka osoba doda toplinu
alpha4 = 0.01; % blagi utjecaj vlažnosti
Troom = alpha0 + alpha1*Tout + alpha2*HVAC + alpha3*Occ + alpha4*RH +
0.5*randn(1,N);

% 2) Sastavi ulaze i izlaze u oblik pogodan za fitnet (dim: [features x
samples])
inputs = [Tout; HVAC; Occ; RH]; % 4 x N
targets = Troom; % 1 x N

% 3) Podjela podataka: 70% train, 15% val, 15% test
trainN = round(0.7*N);
valN = round(0.15*N);
testN = N - trainN - valN;

idx = randperm(N);
train_idx = idx(1:trainN);
```

```

val_idx = idx(trainN+1:trainN+valN);
test_idx = idx(trainN+valN+1:end);

inputs_train = inputs(:,train_idx);
targets_train = targets(:,train_idx);
inputs_val = inputs(:,val_idx);
targets_val = targets(:,val_idx);
inputs_test = inputs(:,test_idx);
targets_test = targets(:,test_idx);

% 4) Kreiranje i treniranje mreže (regresija)
hiddenNeurons = 12;
net = fitnet(hiddenNeurons, 'trainlm'); % Levenberg-Marquardt

% postavi vlastitu podjelu (da MATLAB koristi naše skupove)
net.divideFcn = 'divideind';
net.divideParam.trainInd = train_idx;
net.divideParam.valInd = val_idx;
net.divideParam.testInd = test_idx;

% (opcionalki) regularizacija i tren. parametri
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-4;
net.performParam.regularization = 0.01;

% Treniraj
[net,tr] = train(net, inputs, targets);

% 5) Testiranje
pred_all = net(inputs); % 1 x N (predikcije za sve uzorce)
pred_test = pred_all(:, test_idx);
true_test = targets(:, test_idx);

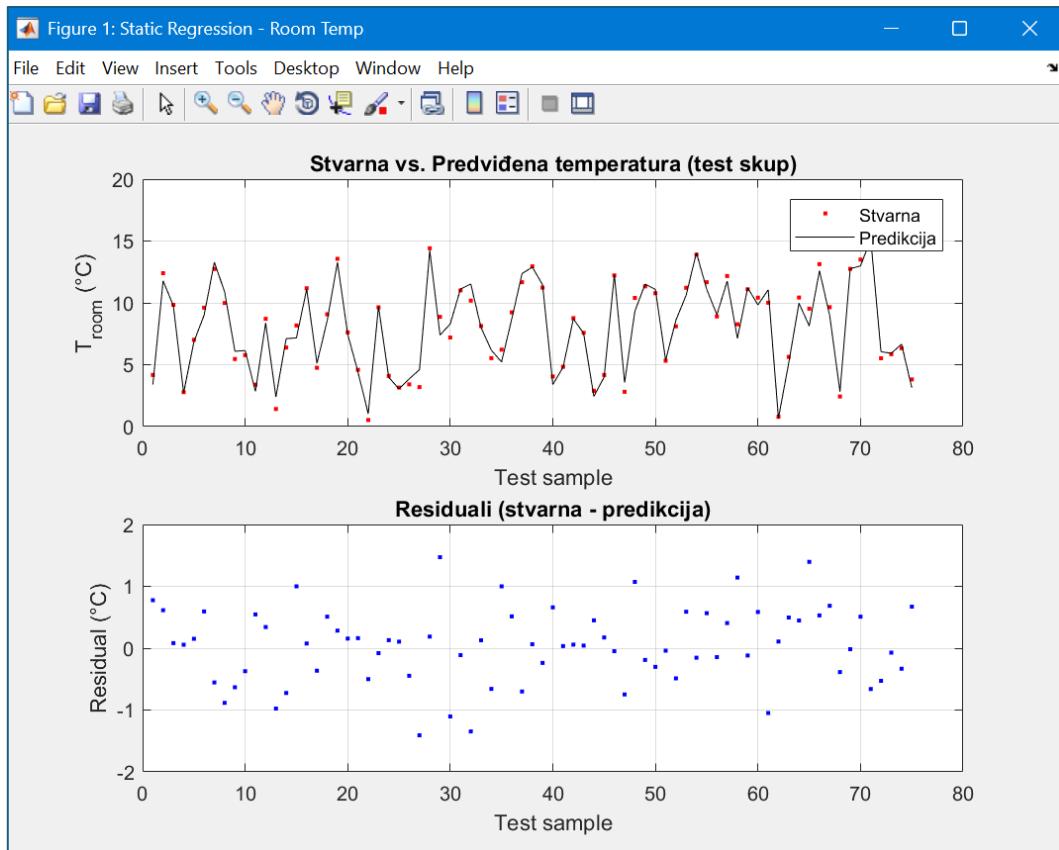
% 6) Vizualizacija
figure('Name','Static Regression - Room Temp');
subplot(2,1,1);
plot(true_test,'r.');?>
plot(pred_test,'k-');
xlabel('Test sample'); ylabel('T_{room} (°C)');
legend('Stvarna','Predikcija');
title('Stvarna vs. Predviđena temperatura (test skup)');
grid on;

subplot(2,1,2);
plot(true_test - pred_test,'b.');
xlabel('Test sample'); ylabel('Residual (°C)');
title('Residuali (stvarna - predikcija)');
grid on;

% 7) MSE i statistika
mse_val = mse(true_test - pred_test);
mae_val = mean(abs(true_test - pred_test));
fprintf('Static model: MSE = %.4f °C^2, MAE = %.4f °C\n', mse_val, mae_val);

```

Static model: MSE = 0.3684 °C^2, MAE = 0.4794 °C



Slika 4

Graf residuala pomaže u pronalaženju pristranosti / sistematskih pogrešaka.

b) Nenadzirano učenje (engl. Unsupervised Learning)

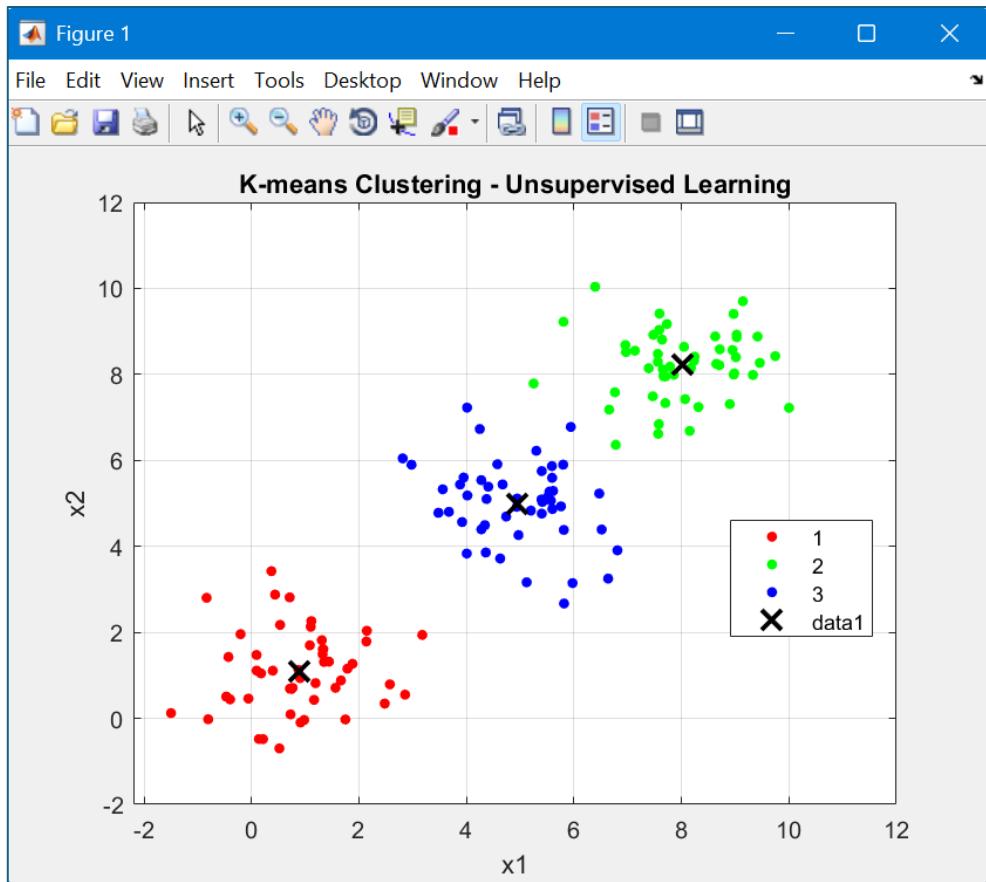
Algoritam uči bez poznatih izlaza, tj. podaci nisu označeni. Cilj je pronaći skrivene strukture u podacima, kao što su grupe ili pravci maksimalne varijacije.

- Cilj: Otkriti uzorke, klastere ili latentne dimenzije u podacima.
- Primjene: Grupisanje tipova potrošača energije, analiza radnih režima sistema.
- Algoritmi: K-means, PCA, Autoencoder.

Primjer5

```
% Nenadzirano učenje - K-means klasterovanje
data = [randn(50,2)+1; randn(50,2)+5; randn(50,2)+8];
[idx, C] = kmeans(data, 3);

gscatter(data(:,1), data(:,2), idx);
hold on;
plot(C(:,1), C(:,2), 'kx', 'MarkerSize', 12, 'LineWidth', 2);
title('K-means Clustering - Unsupervised Learning');
xlabel('x1'); ylabel('x2');
grid on;
```



Slika 5

Dobiveni podaci su grupisani u tri klastera, a centri klastera su označeni crnim „X“. Ovim se pokazuje sposobnost algoritma da sam pronađe obrasce bez unaprijed poznatih oznaka.

Primjer6 Na osnovu dnevne potrošnje električne energije u domaćinstvima, sistem treba automatski da grapiše potrošače u kategorije:

- potrošači s malom potrošnjom,
- potrošači sa srednjom potrošnjom,
- potrošači s velikom potrošnjom.

*Nema unaprijed poznatih oznaka (etiketa) — sistem sam otkriva obrasce u podacima. Nenadzirano učenje se koristi kada podaci nemaju označene izlaze. Cilj je pronaći strukturu ili sličnost između uzoraka. Najpoznatija metoda je K-means clustering — algoritam koji dijeli podatke u K grupa (klastera) na osnovu međusobne sličnosti.

```
%% Primjer 5: Nenadzirano učenje – K-means clustering podataka o potrošnji
%% električne energije
% MATLAB R2017b kompatibilno

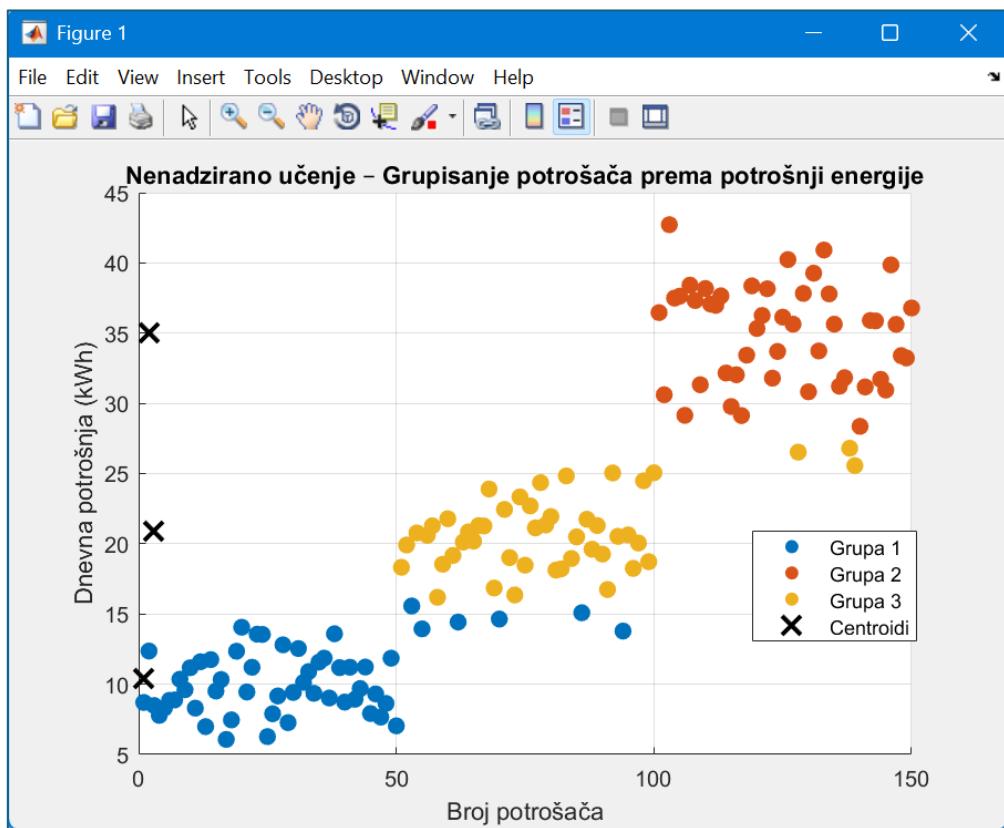
% Generisanje podataka (npr. potrošnja u kWh tokom dana)
rng(1); % zbog ponovljivosti rezultata
data = [randn(50,1)*2 + 10; randn(50,1)*3 + 20; randn(50,1)*4 + 35];
% data = 150 vrijednosti koje predstavljaju potrošnju energije (kWh)

% Broj grupa (klastera)
K = 3;

% Primjena K-means algoritma
[idx, C] = kmeans(data, K);
```

```
% Prikaz rezultata
figure;
hold on;
colors = lines(K);
for i = 1:K
    scatter(find(idx==i), data(idx==i), 60, colors(i,:), 'filled');
end
plot(C, 'kx', 'MarkerSize', 12, 'LineWidth', 2);
xlabel('Broj potrošača');
ylabel('Dnevna potrošnja (kWh)');
title('Nenadzirano učenje - Grupisanje potrošača prema potrošnji energije');
legend('Grupa 1','Grupa 2','Grupa 3','Centroidi');
grid on;

% Analiza grupe
disp('Prosječna potrošnja po grupi (kWh):');
disp(C);
```



Slika 6

Prosječna potrošnja po grupi (kWh):

10.4161

35.0019

20.9174

Primjer7 Grupisati radne tačke solarnog panela (*Sunčeva radijacija (G)* [W/m²], *Temperatura panela (T)* [°C], *Napon (V)* [V] i *Struja (I)* [A]) kako bi se definisali radni režimi bez unaprijed poznatih izlaza. Ulazni podaci predstavljaju mjerne vrijednosti prikupljene u različitim vremenskim uslovima rada solarnog panela.

```
clear; clc; close all;

%% 1. Ulazni podaci
% Svaka vrsta predstavlja mjerne podatke PV panela u određenim uslovima
Radiation = [600; 800; 1000; 400; 300; 950; 700; 850; 200; 500];
Temperature = [20; 25; 28; 18; 15; 27; 22; 26; 14; 19];
Voltage = [10.5; 11.8; 12.0; 9.2; 8.5; 11.9; 10.8; 11.5; 7.8; 9.5];
Current = [3.2; 4.0; 4.5; 2.8; 2.0; 4.3; 3.6; 3.9; 1.5; 2.7];

% Kreiranje matrice podataka
Data = [Radiation Temperature Voltage Current];

%% 2. Broj klastera
numClusters = 3;

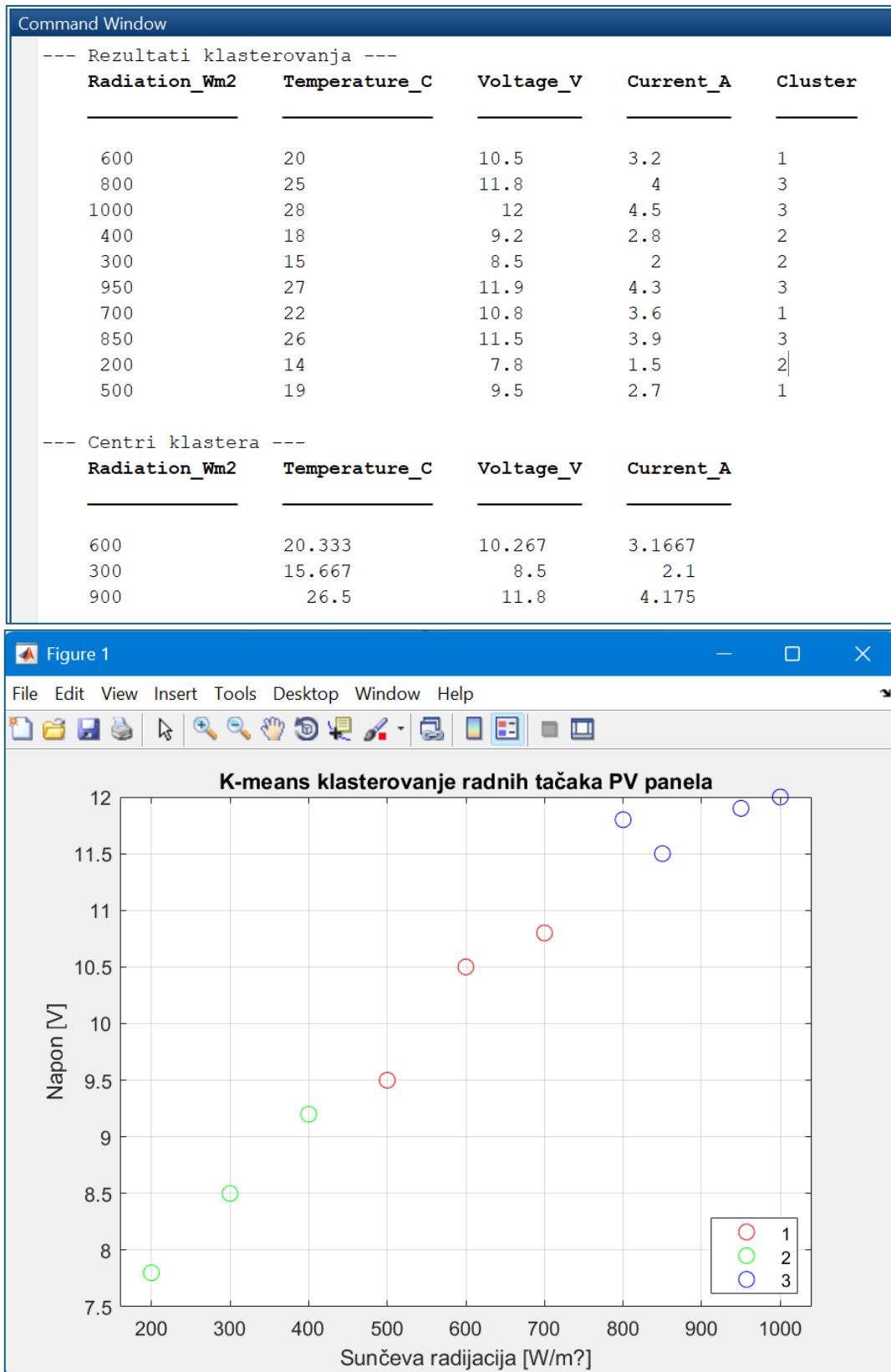
%% 3. K-means klasterovanje
[cluster_idx, cluster_centers] = kmeans(Data, numClusters, 'Replicates', 10);

%% 4. Prikaz rezultata
disp('--- Rezultati klasterovanja ---');
Results = table(Radiation, Temperature, Voltage, Current, cluster_idx, ...
    'VariableNames',
    {'Radiation_Wm2','Temperature_C','Voltage_V','Current_A','Cluster'});
disp(Results);

%% 5. Grafički prikaz (radijacija vs. napon)
figure;
gscatter(Radiation, Voltage, cluster_idx, 'rgb', 'o', 8);
xlabel('Sunčeva radijacija [W/m?]');
ylabel('Napon [V]');
title('K-means klasterovanje radnih tačaka PV panela');
grid on;

%% 6. Prikaz centara klastera
disp('--- Centri klastera ---');
disp(array2table(cluster_centers, ...
    'VariableNames',
    {'Radiation_Wm2','Temperature_C','Voltage_V','Current_A'}));

%% 7. Interpretacija (komentar)
% Klaster 1 -> Niska radijacija i napon -> neefikasan režim rada
% Klaster 2 -> Umjerena radijacija i stabilan napon -> normalni radni režim
% Klaster 3 -> Visoka radijacija i visoka efikasnost -> optimalni radni režim
```



Slika 7

Primjer8 Dati su podaci o temperaturi, vlažnosti zraka, vlažnosti tla, osvijetljenosti i koncentraciji CO₂ koji se prikupljaju pomoću senzora (npr. DHT22, soil moisture sensor, BH1750, MQ135). Napisati kod koji će omogućiti analiziranje kako bi se prepoznale grupe (klasteri) radnih uslova:

- optimalni uslovi,
- pregrijavanje,
- suvi ili zasićeni uslovi.

```
clear; clc; close all;

%% 1. Simulirani senzorski podaci (npr. mjerenja u toku dana)
% temperature [°C], humidity [%], soil moisture [%], light [lux], CO2
[ppm]
Temperature = [22 24 28 35 18 20 27 30 33 25 26 21 19 32 29]';
Humidity = [60 55 50 40 80 75 52 48 45 58 54 70 72 42 49]';
SoilMoisture = [40 35 30 25 60 55 38 32 28 36 33 58 57 29 31]';
Light = [15000 20000 35000 40000 10000 12000 25000 37000 42000 21000 23000
14000 11000 38000 36000]';
CO2 = [450 480 600 720 420 430 580 700 750 500 520 460 440 680 640]';

% formiraj dataset
Data = [Temperature Humidity SoilMoisture Light CO2];

%% 2. K-means klasterovanje
numClusters = 3;
[cluster_idx, cluster_centers] = kmeans(Data, numClusters, 'Replicates',
10);

%% 3. Rezultati
disp('--- Rezultati klasterovanja ---');
Results = table(Temperature, Humidity, SoilMoisture, Light, CO2,
cluster_idx, ...
    'VariableNames',
{'Temp_C','Humidity','SoilMoisture','Light_lux','CO2_ppm','Cluster'});
disp(Results);

%% 4. Prikaz klastera (npr. Temperatura vs. Vlažnost)
figure;
gscatter(Temperature, Humidity, cluster_idx, 'rgb', 'o', 8);
xlabel('Temperatura [°C]');
ylabel('Relativna vlažnost zraka [%]');
title('K-means analiza uslova u plasteniku');
grid on;

%% 5. Centri klastera
disp('--- Centri klastera ---');
disp(array2table(cluster_centers, ...
    'VariableNames',
{'Temp_C','Humidity','SoilMoisture','Light_lux','CO2_ppm'}));

%% 6. Interpretacija (komentar)
% Klaster 1 -> Optimalni uslovi (umjereni temperaturni i vlagi)
% Klaster 2 -> Pregrijavanje i niska vlagi -> potreba za ventilacijom i
zalijevanjem
% Klaster 3 -> Niska temperaturna, visoka vlagi -> rizik od kondenzacije i
gljivica
```

Editor - C:\Users\unaed\OneDrive\Desktop\AI_MATERIJAL\matlab_primeri\nenadrzirano_pr5.m

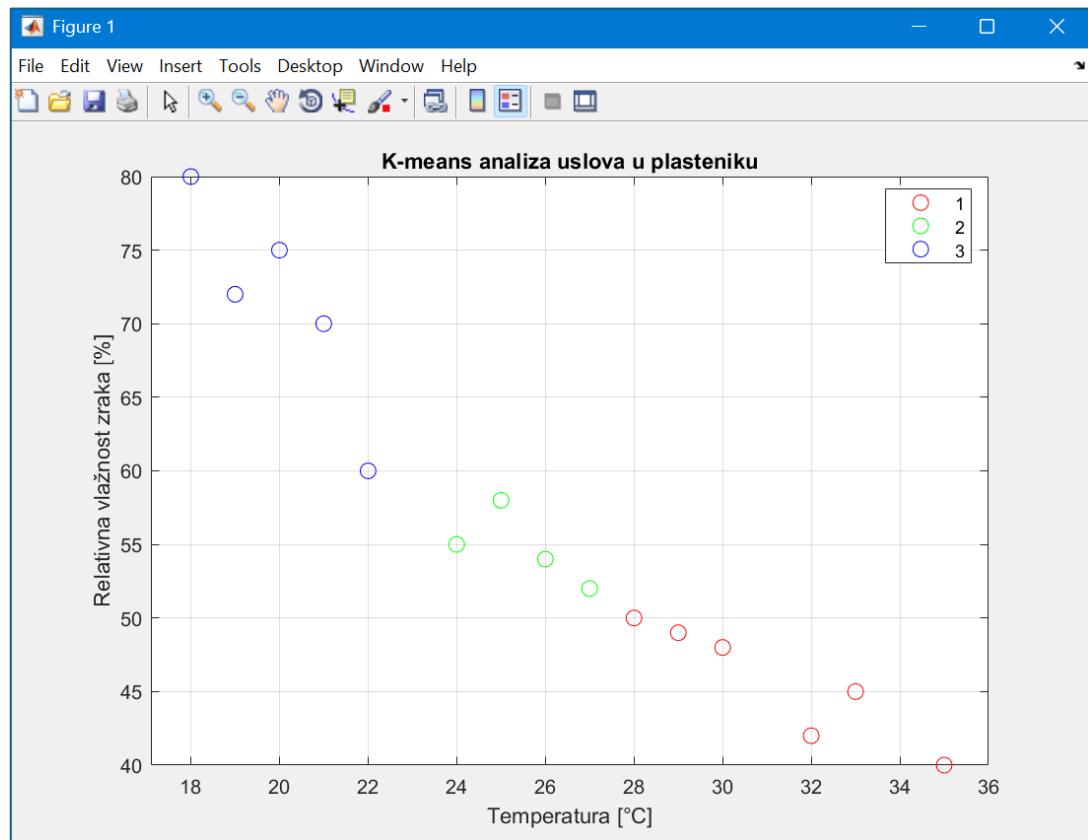
Command Window

--- Rezultati klasterovanja ---

Temp_C	Humidity	SoilMoisture	Light_lux	CO2_ppm	Cluster
22	60	40	15000	450	3
24	55	35	20000	480	2
28	50	30	35000	600	1
35	40	25	40000	720	1
18	80	60	10000	420	3
20	75	55	12000	430	3
27	52	38	25000	580	2
30	48	32	37000	700	1
33	45	28	42000	750	1
25	58	36	21000	500	2
26	54	33	23000	520	2
21	70	58	14000	460	3
19	72	57	11000	440	3
32	42	29	38000	680	1
29	49	31	36000	640	1

--- Centri klastera ---

Temp_C	Humidity	SoilMoisture	Light_lux	CO2_ppm
31.167	45.667	29.167	38000	681.67
25.5	54.75	35.5	22250	520
20	71.4	54	12400	440

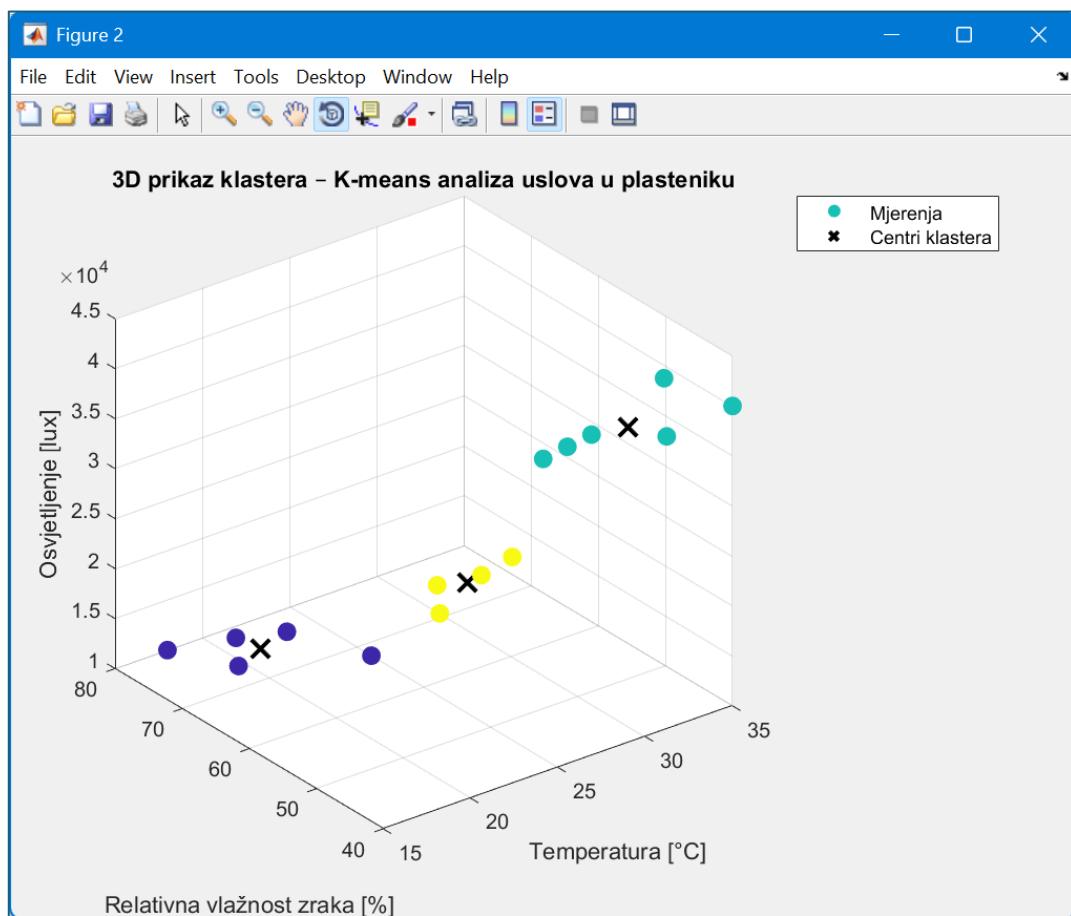


Slika 8

```

%% 3D prikaz klastera
figure;
scatter3(Temperature, Humidity, Light, 80, cluster_idx, 'filled');
hold on;
scatter3(cluster_centers(:,1), cluster_centers(:,2), cluster_centers(:,4),
150, 'kx', 'LineWidth', 2);
xlabel('Temperatura [°C]');
ylabel('Relativna vlažnost zraka [%]');
zlabel('Osvjetljenje [lux]');
title('3D prikaz klastera - K-means analiza uslova u plasteniku');
legend('Mjerenja', 'Centri klastera');
grid on;
rotate3d on;

```



Slika 9

c) Polunadzirano učenje (engl. Semi-Supervised Learning)

Ovdje algoritam koristi kombinaciju označenih i neoznačenih podataka.

Praktično, u realnim sistemima često je teško označiti sve podatke, pa model uči i iz neoznačenih primjera.

- Cilj: Poboljšati učenje korištenjem velike količine neoznačenih podataka zajedno s malim brojem označenih.
- Primjene: Detekcija grešaka u proizvodnji, analiza potrošnje energije, predikcija kvarova u sistemima.
- Algoritmi: Semi-supervised SVM, Label Propagation, Self-training.

Primjer9 Zadatak je predvidjeti optimalno stanje ventilacije u plasteniku (otvorena/zatvorena), koristeći djelimično označene podatke prikupljene senzorima:

- Temperatura zraka (°C)
- Vlažnost zraka (%)
- Vlažnost tla (%)
- Intenzitet svjetla (lux)

Od ukupnog skupa podataka, samo dio (npr. 20%) je ručno označen kao:

- 1 = ventilacija treba biti uključena
- 0 = ventilacija treba biti isključena

Ostatak podataka nema oznaku, model treba sam naučiti obrasce i dopuniti oznake koristeći semi-supervised learning.

```
%% Polunadzirano ucenje - ventilacija u plasteniku
clc; clear; close all;

%% Simulacija determinističkih podataka (24 sata)
n = 100;
time = linspace(0, 24, n)';

Temperature = 20 + 10*sin(time/24*2*pi); % °C
Humidity = 50 + 20*sin(time/24*2*pi + 1); % %
Light = 1000 + 800*sin(time/24*2*pi - 1); % lux

%% Definicija stvarnog stanja ventilacije
Ventilation = double(Temperature > 28 | Humidity > 70 | Light > 1800);

%% Polunadzirano ucenje - svi podaci označeni
Labels = Ventilation;
X = [Temperature, Humidity, Light];

%% Treniranje decision tree modela (složen)
trainedModel = fitctree(X, Labels, 'MaxNumSplits', 50);

%% Predikcija ventilacije
Predicted = predict(trainedModel, X);

%% Dodajemo minimalno odstupanje za vizualizaciju
epsilon = 0.05; % minimalna razlika
Predicted_plus = min(Predicted + epsilon, 1);
Predicted_minus = max(Predicted - epsilon, 0);
```

```

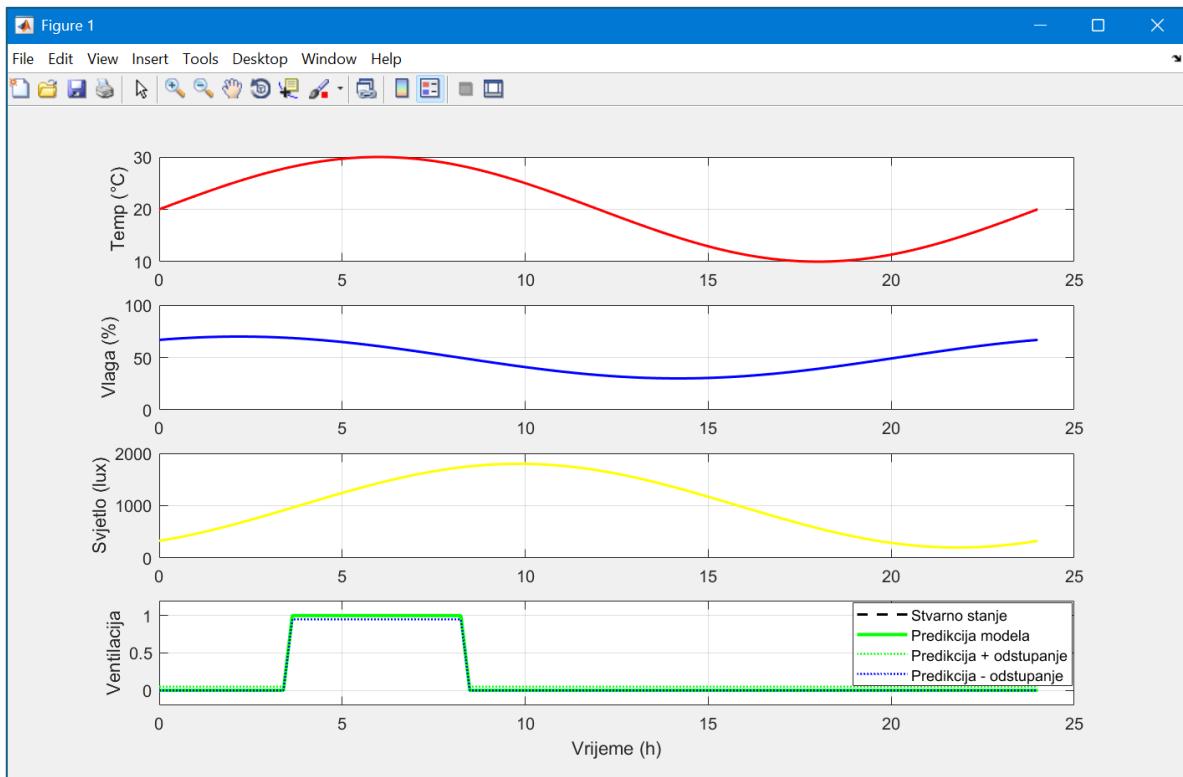
%% Linijski graf
figure;

subplot(4,1,1)
plot(time, Temperature, '-r', 'LineWidth', 1.5); ylabel('Temp (°C)'); grid on;
subplot(4,1,2)
plot(time, Humidity, '-b', 'LineWidth', 1.5); ylabel('Vлага (%)'); grid on;
subplot(4,1,3)
plot(time, Light, '-y', 'LineWidth', 1.5); ylabel('Svjetlo (lux)'); grid on;

subplot(4,1,4)
plot(time, Ventilation, '--k', 'LineWidth', 1.5, 'DisplayName', 'Stvarno stanje');
hold on;
plot(time, Predicted, '-g', 'LineWidth', 2, 'DisplayName', 'Predikcija modela');
plot(time, Predicted_plus, ':g', 'LineWidth', 1.2, 'DisplayName', 'Predikcija + odstupanje');
plot(time, Predicted_minus, ':b', 'LineWidth', 1.2, 'DisplayName', 'Predikcija - odstupanje');
xlabel('Vrijeme (h)'); ylabel('Ventilacija');
ylim([-0.2 1.2]);
legend('Location','best'); grid on;

% Naslov figure
axes('Position',[0 0 1 1], 'Visible', 'off');
text(0.5, 1, 'Polunadzirano učenje - Minimalno odstupanje predikcije i stvarnog stanja', ...
'HorizontalAlignment', 'center', 'VerticalAlignment', 'top', 'FontSize', 12, 'FontWeight', 'bold');


```



Slika 10

```

%% Graf sa bojama
figure; hold on;

% Stvarno stanje
area(time, Ventilation, 'FaceColor',[0.8 0.8 0.8], 'DisplayName','Stvarno stanje');

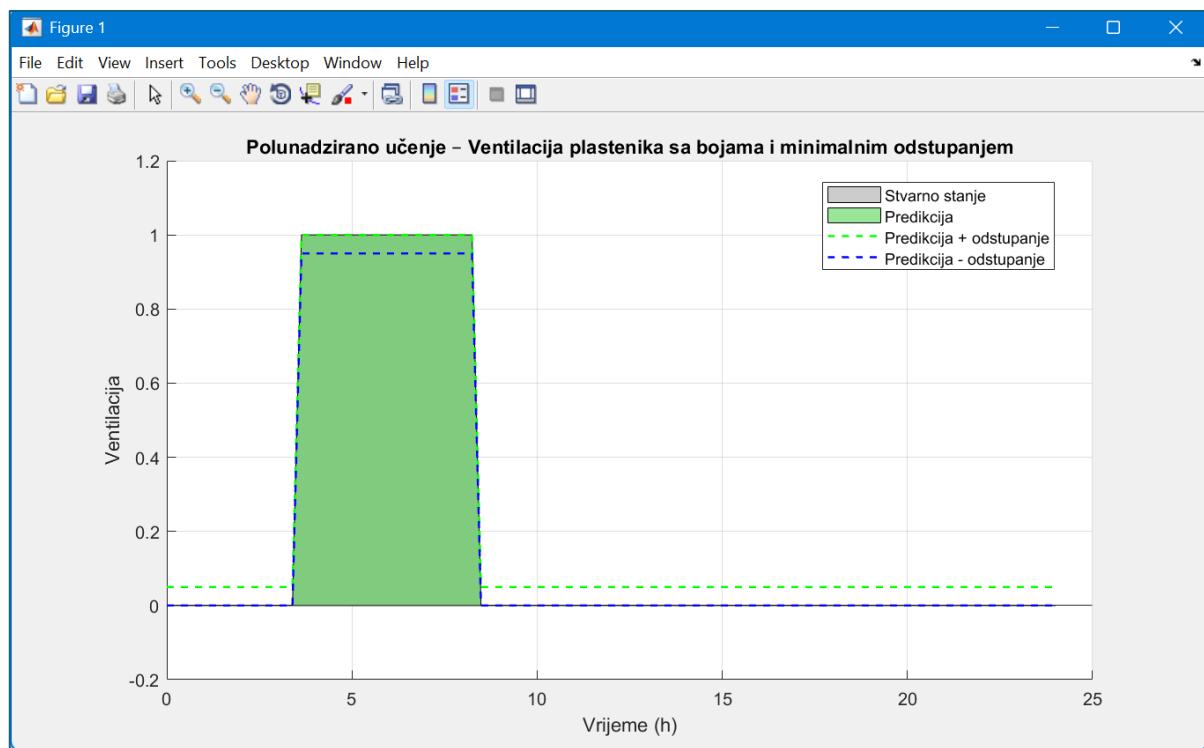
% Predikcija
area(time, Predicted, 'FaceColor',[0.2 0.8 0.2], 'FaceAlpha',0.5,
'DisplayName','Predikcija');

% Minimalno odstupanje
plot(time, Predicted_plus, '--g', 'LineWidth',1.2, 'DisplayName','Predikcija + odstupanje');
plot(time, Predicted_minus, '--b', 'LineWidth',1.2, 'DisplayName','Predikcija - odstupanje');

xlabel('Vrijeme (h)');
ylabel('Ventilacija');
ylim([-0.2 1.2]);
grid on;
legend('Location','best');

% Naslov figure
title('Polunadzirano učenje – Ventilacija plastenika sa bojama i minimalnim odstupanjem');

```



Slika 11

UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

DUBOKO UČENJE (DEEP LEARNING)

Konvolucijska neuronska mreža (CNN) je poseban tip duboke mreže koja je efikasna za obradu 2D podataka (slike). Ključne osobine:

- **Konvolucijski slojevi** (filteri) automatski uče lokalne obrasce (rubove, teksture).

- **Pooling slojevi** smanjuju dimenzije i čuvaju značajke.

- **Fully connected** i **softmax** slojevi mapiraju izdvojene značajke na klase.

MATLAB pruža visoke apstrakcije (layers, datastores, trainingOptions, trainNetwork) koje ubrzavaju razvoj i eksperimentiranje.

Učitavanje i priprema podataka

imageDatastore — čita slike iz foldera i automatski pravi label-e iz imena foldera.

```
imds = imageDatastore('path/to/data', 'IncludeSubfolders', true,  
'LabelSource', 'foldernames');
```

splitEachLabel — podjela po klasama (train/val/test):

```
[imdsTrain, imdsVal, imdsTest] = splitEachLabel(imds, 0.7, 0.15, 0.15,  
'randomize');
```

augmentedImageDatastore — skaliranje/augmentacija pri učitavanju (korisno za pretrained mreže koje očekuju specifične dimenziјe):

```
augImdsTrain = augmentedImageDatastore([224 224], imdsTrain,  
'ColorPreprocessing', 'gray2rgb');
```

imageDataAugmenter — definiranje augmentacija (rotacija, flip, shift):

```
augmenter = imageDataAugmenter('RandRotation', [-10 10], 'RandXTranslation', [-3  
3], 'RandYReflection', true);  
augImds = augmentedImageDatastore([28 28], imdsTrain, 'DataAugmentation',  
augmenter);
```

Primjer1

Dataset sadrži slike rukom pisanih cifara 0–9. **imageDatastore** je poseban MATLAB objekat koji čita slike iz foldera i automatski im dodjeljuje oznake (labels) prema nazivu foldera. Svaka slika će biti povezana s klasom (0–9) prema folderu u kojem se nalazi.

```
close all  
clear all  
clc  
  
%ucitavanje slika, svaka slika je gray skala 28x28 piksela  
digitDatasetPath = fullfile(matlabroot, 'toolbox', 'nnet', 'nnndemos', ...  
'nndatasets', 'DigitDataset');  
imds = imageDatastore(digitDatasetPath, ...  
'IncludeSubfolders', true, 'LabelSource', 'foldernames');  
  
% test/train/validation  
% 60% trening, 20%validacija i 20% testiranje  
fracTrainFiles = 0.6;  
fracValFiles = 0.2;  
fracTestFiles = 0.2;
```

```

[imdsTrain, imdsValidation, imdsTest] = splitEachLabel(imds, ...
    fracTrainFiles, fracValFiles, fracTestFiles, 'randomize');

% definiranje CNN arhitekture
%Struktura:
%Input layer: ulaz 28×28×1 (grayscale slika).
%Conv layer: 10 filtera, kernel 3×3, padding 'same'.
%Batch norm: stabilizira učenje.
%Max pooling: smanjuje dimenzije slike (downsampling).
%Drugi conv + ReLU: dodaje nelinearnost i složenije reprezentacije.
%Fully connected layer (10): mapira na 10 klasa (cifre 0-9).
%Softmax: pretvara rezultate u vjerovatnoće.
%Classification layer: koristi cross-entropy gubitak.

layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,10,'Padding','same')
    batchNormalizationLayer
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,10,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

%% Stochastic Gradient Descent with Momentum (SGDM)
% Početna brzina učenja: 0.01
%100 epoha
% Validacija svakih 30 iteracija
% Prikazuje training-progress graf

options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',100, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');

net = trainNetwork(imdsTrain,layers,options);

%%
YPred = classify(net,imdsTest);
YTest = imdsTest.Labels;

% broj tacno klasificiranih/boj test uzoraka
accuracy = sum(YPred == YTest)/numel(YTest)

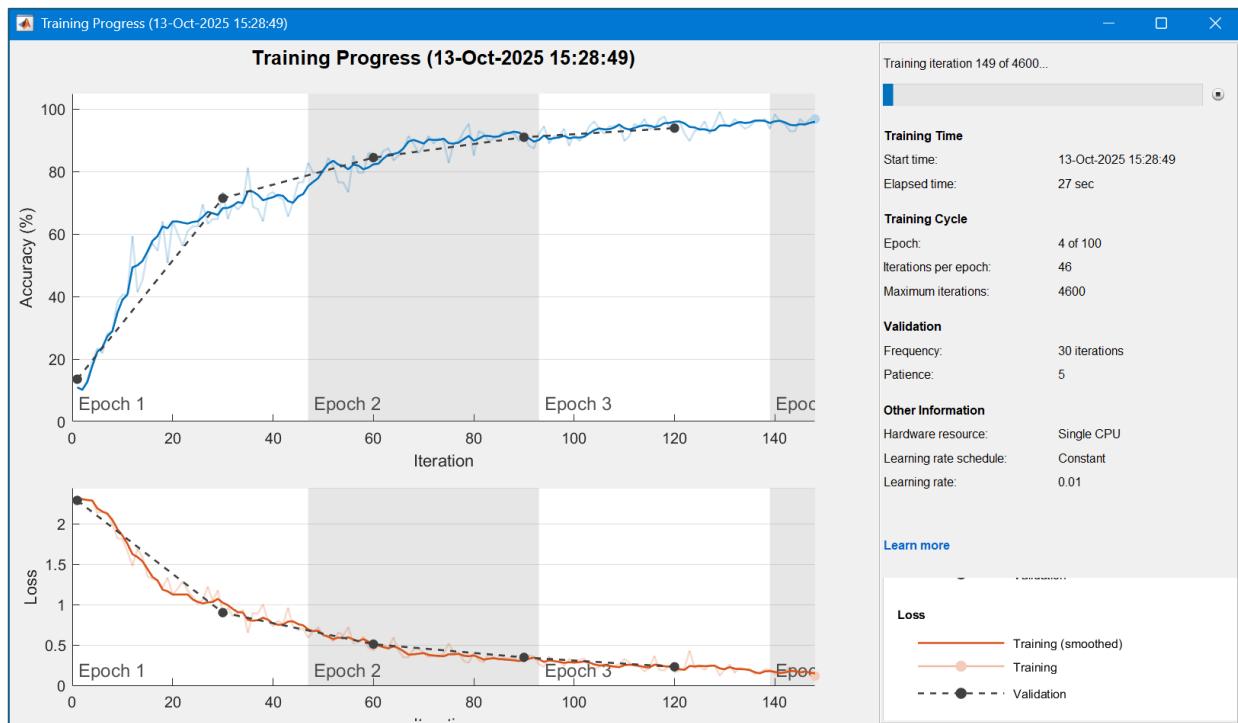
%%
ind = find(YPred ~= YTest);
figure;
for ii = 1:9
    subplot(3,3,ii);
    imagesc(imdsValidation.readimage(ind(ii)));

```

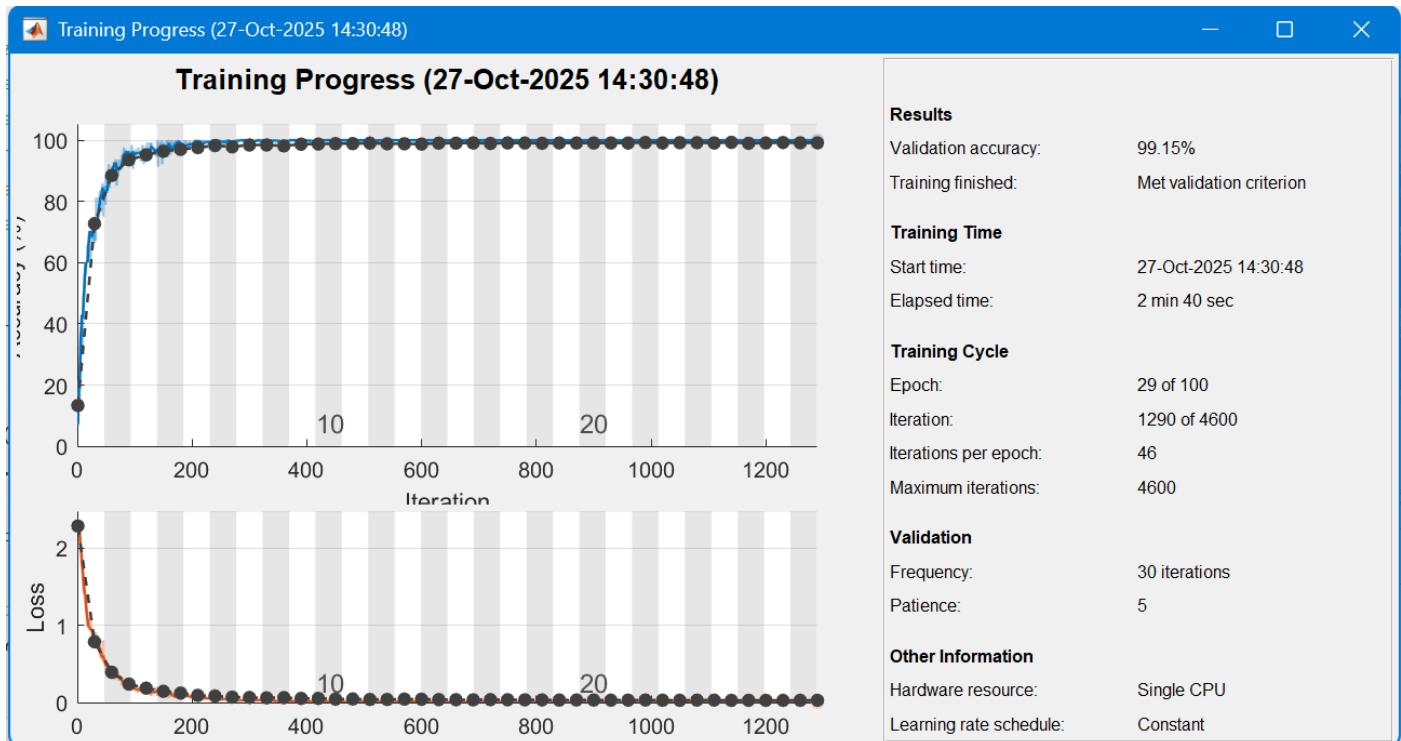
```

        title([num2str(double(YPred(ind(ii)))-1), ' predicted, ', ...
        num2str(double(YTest(ind(ii)))-1), ' actual'])
end

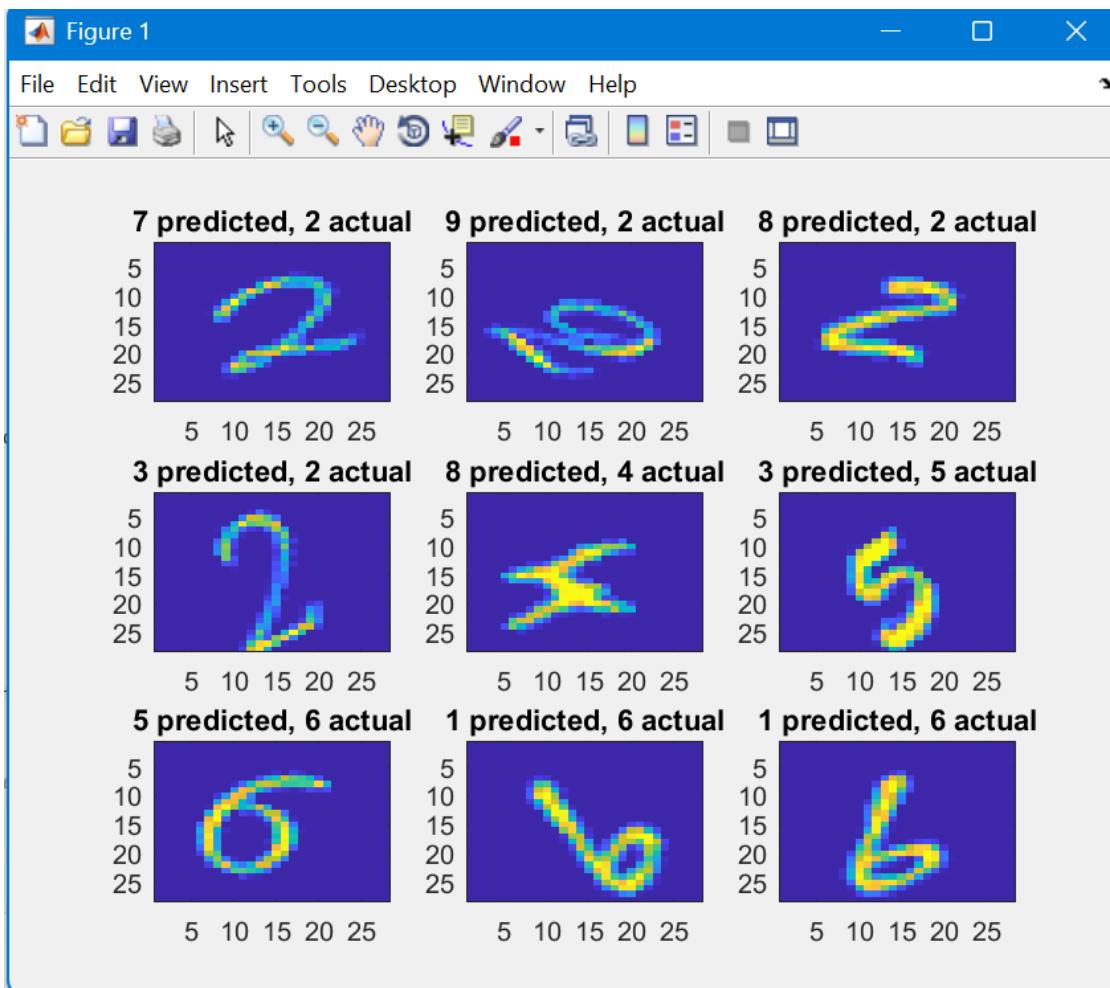
```



Slika 1



Slika 2



Slika 3

Primjer2

Potrebno je napisati program koji prepozna znakove sa registrarske tablice vozila primjenom metoda obrade slike i poređenja šablona (template matching). Potrebno je implementirati kod koji učitava sliku vozila sa vidljivom registrarskom tablicom, izdvaja područje registrarske tablice iz slike korištenjem morfoloških operacija i detekcije ivica, razdvaja pojedinačne karaktere (slova i brojeve) sa tablice, prepozna svaki karakter poređenjem sa unaprijed definisanim šablonima i prikazuje prepoznatu registrarsku oznaku na ekranu.

Rješenje:

Skripta	Funkcija	Šta radi
Create_templates.m	Kreira bazu poznatih znakova	Učitava slike slova/brojeva, spremi ih u .mat datoteku
readLetter.m	Prepozna jedan znak	Poredi ulaznu sliku znaka sa bazom pomoću korelacije
numberplate.m	Glavni program	Izdvaja tablicu sa slike, prepozna znakove, sklapa registrarski broj

Create_templates.m kreira bazu poznatih slika (slova i brojeva) koje se koriste kao referenca za poređenje.

Create_templates.m

```
%CREATE TEMPLATES
%Letter
A=imread('char/A.bmp');B=imread('char/B.bmp');
C=imread('char/C.bmp');D=imread('char/D.bmp');
E=imread('char/E.bmp');F=imread('char/F.bmp');
G=imread('char/G.bmp');H=imread('char/H.bmp');
I=imread('char/I.bmp');J=imread('char/J.bmp');
K=imread('char/K.bmp');L=imread('char/L.bmp');
M=imread('char/M.bmp');N=imread('char/N.bmp');
O=imread('char/O.bmp');P=imread('char/P.bmp');
Q=imread('char/Q.bmp');R=imread('char/R.bmp');
S=imread('char/S.bmp');T=imread('char/T.bmp');
U=imread('char/U.bmp');V=imread('char/V.bmp');
W=imread('char/W.bmp');X=imread('char/X.bmp');
Y=imread('char/Y.bmp');Z=imread('char/Z.bmp');
Afill=imread('char/fillA.bmp');
Bfill=imread('char/fillB.bmp');
Dfill=imread('char/fillD.bmp');
Ofill=imread('char/fillO.bmp');
Pfill=imread('char/fillP.bmp');
Qfill=imread('char/fillQ.bmp');
Rfill=imread('char/fillR.bmp');

%iz foldera char ucitavaju se slike svih karaktera A-Z i 0-9 u formatu .bmp

%Number
one=imread('char/1.bmp'); two=imread('char/2.bmp');
three=imread('char/3.bmp');four=imread('char/4.bmp');
five=imread('char/5.bmp'); six=imread('char/6.bmp');
seven=imread('char/7.bmp');eight=imread('char/8.bmp');
nine=imread('char/9.bmp'); zero=imread('char/0.bmp');
zerofill=imread('char/fill0.bmp');
fourfill=imread('char/fill4.bmp');
sixfill=imread('char/fill6.bmp');
sixfill2=imread('char/fill6_2.bmp');
eightfill=imread('char/fill8.bmp');
ninefill=imread('char/fill9.bmp');
ninefill2=imread('char/fill9_2.bmp');

%Creating the array of aplhabets
letter=[A Afill B Bfill C Cfill D Dfill E Efill F Ffill G Gfill H Hfill I Ifill J Jfill K Kfill L Lfill M Mfill N Nfill O Ofill P Pfill Q Qfill R Rfill S Sfill T Tfill U Ufill V Vfill W Wfill X Xfill Y Yfill Z Zfill];
%Creating the array of numbers sve slike se spajaju u jedan niz
number=[one two three four fourfill five six sixfill sixfill2 seven eight eightfill nine ninefill ninefill2 zero zerofill];

character=[letter number];
% svaka ćelija cell sadrži jednu sliku velicine 42x24 piksela
NewTemplates=mat2cell(character,42,[24 24 24 24 24 24 24 24 24 ...]
```

```

24 24 24 24 24 24 24 ...
24 24 24 24 24 24 24 ...
24 24 24 24 24 24 24 ...
24 24 24 24 24 24 24 ...
24 24 24 24 24 24 24 ...
24 24 24 24 24 24 24];
save ('NewTemplates','NewTemplates')
clear all

```

Sljedeci m.file se koristi da primi binarnu sliku jednog znaka i vraca koji je to znak, slovo ili broj

readLetter.m

```

function letter=readLetter(snap)
%READLETTER reads the character from the character's binary image.
% LETTER=READLETTER(SNAP) outputs the character in class 'char' from the
% input binary image SNAP.

load NewTemplates % Loads the templates of characters in the memory.
snap=imresize(snap,[42 24]); % Resize the input image so it can be compared with the template's
images.
comp=[];
for n=1:length(NewTemplates)
% corr2 racuna koeficijent korelacije izmedu slike znaka i svakog od šablonu
% vrijednost ide -1 1, sto je veca bliza je 1, slike su sličnije.
sem=corr2(NewTemplates{1,n},snap); % Correlation the input image with every image in the
template for best matching.
comp=[comp sem]; % Record the value of correlation for each template's character.
%display(sem);

end
%vd je indeks znaka iz baze koji najvise lici na ulazni znak
vd=find(comp==max(comp)); % Find the index which correspond to the highest matched character.
%display(max(comp));
%*-_*-*-*-*-*-*-*-*-*-*-
% According to the index assign to 'letter'.
% Alphabets listings.
% if elseif blok odreduje koji karakter odgovara tom indeksu jer vise šablonu moze biti isti znak
% na kraju funkcije treba da vraca prepoznati znak u obliku stringa npr. 'A' '5' ...
if vd==1 || vd==2
letter='A';
elseif vd==3 || vd==4
letter='B';
elseif vd==5
letter='C';
elseif vd==6 || vd==7
letter='D';
elseif vd==8
letter='E';
elseif vd==9

```

```
letter='F';
elseif vd==10
    letter='G';
elseif vd==11
    letter='H';
elseif vd==12
    letter='I';
elseif vd==13
    letter='J';
elseif vd==14
    letter='K';
elseif vd==15
    letter='L';
elseif vd==16
    letter='M';
elseif vd==17
    letter='N';
elseif vd==18 || vd==19
    letter='O';
elseif vd==20 || vd==21
    letter='P';
elseif vd==22 || vd==23
    letter='Q';
elseif vd==24 || vd==25
    letter='R';
elseif vd==26
    letter='S';
elseif vd==27
    letter='T';
elseif vd==28
    letter='U';
elseif vd==29
    letter='V';
elseif vd==30
    letter='W';
elseif vd==31
    letter='X';
elseif vd==32
    letter='Y';
elseif vd==33
    letter='Z';
%*-*-*-*
% Numerals listings.
elseif vd==34
    letter='1';
elseif vd==35
    letter='2';
elseif vd==36
    letter='3';
elseif vd==37 || vd==38
    letter='4';
elseif vd==39
```

```

letter='5';
elseif vd==40 || vd==41 || vd==42
    letter='6';
elseif vd==43
    letter='7';
elseif vd==44 || vd==45
    letter='8';
elseif vd==46 || vd==47 || vd==48
    letter='9';
else
    letter='0';
end
end

```

Sljedeci m.file ima za cilj da izdvoji i prepozna registarsku tablicu sa slike automobila/vozila.

numberplate.m

```

clc;
close all;
clear all;

%ucitavanje slike
%im = imread('Images/audi.jpg');
im = imread('Images/6.jpg');
%im = imread('Images/car1.PNG');

%resize slike
im = imresize(im, [480 NaN]);

%prikaz originalne slike
imshow(im),title("Originalna reg. tabla");

%pretvori u grayscale
imgray = rgb2gray(im);

%pretvori u binarnu
imbin = imbinarize(imgray);

%detekcija rubova korištenjem Sobelovog algoritma
im = edge(imgray, 'sobel');

%proširuje sliku koristeći dijamantsku strukturu
im = imdilate(im, strel('diamond', 2));

%fills holes by making it full white color
im = imfill(im, 'holes');

%extract the solid filled parts
im = imerode(im, strel('diamond', 10));

%calculates Area,Bounding Box and Image size
Iprops=regionprops(im,'BoundingBox','Area', 'Image');

%stores the area of the first element

```

```

area = Iprops.Area;
%counts the number of elements
count = numel(Iprops);
%variable to store max area, initial value=first element's area
maxa= area;
%stores bounding box
boundingBox = Iprops.BoundingBox;
%loop to find the bounding box with the maximum area
for i=1:count
    if maxa<Iprops(i).Area
        maxa=Iprops(i).Area;
        boundingBox=Iprops(i).BoundingBox;
    end
end

%all above steps are to find location of the number plate
%Now crop the binarized image to get the number plate only
im = imcrop(imbin, boundingBox);

%resize number plate to 240 NaN
im = imresize(im, [240 NaN]);

%clear dust
im = imopen(im, strel('rectangle', [4 4]));

%remove some object if it's width is too long or too small than 500
im = bwareaopen(~im, 500);

%get width
[h, w] = size(im);

figure;
imshow(im),title("Izdvojena reg. tabla sa izolovanim karakterima");

% Read letter
Iprops=regionprops(im,'BoundingBox','Area', 'Image');
count = numel(Iprops);

noPlate=[]; % Initializing the variable of number plate string.

hold on; % this is used to currunt plot active fro green box
for i=1:count % this fo loop is used to give the green border to
extracted number plate
    ow = length(Iprops(i).Image(1,:));
    oh = length(Iprops(i).Image(:,1));
    if ow<(h/2) && oh>(h/3)
        rectangle('Position', Iprops(i).BoundingBox, 'EdgeColor', 'g',
'LineWidth', 2);
    end
end
for i=1:count
    %width of the ith character of number plate
    ow = length(Iprops(i).Image(1,:));
    %length of the ith character of number plate
    oh = length(Iprops(i).Image(:,1));
    if ow<(h/2) && oh>(h/3)

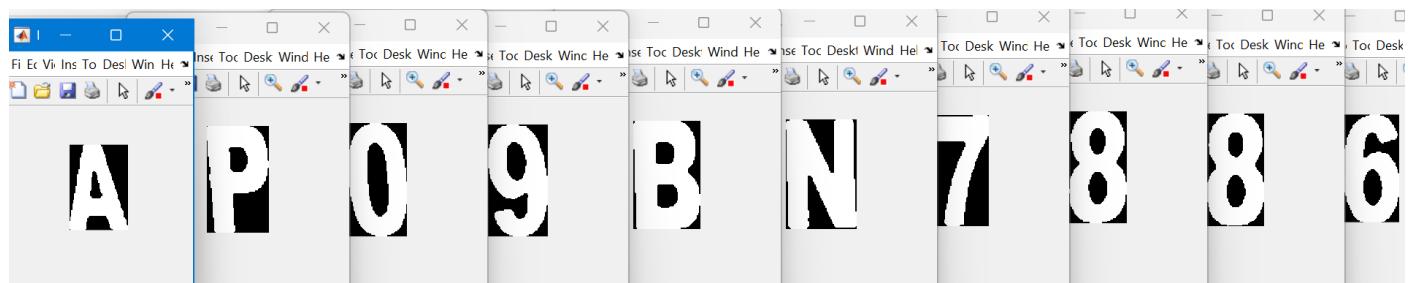
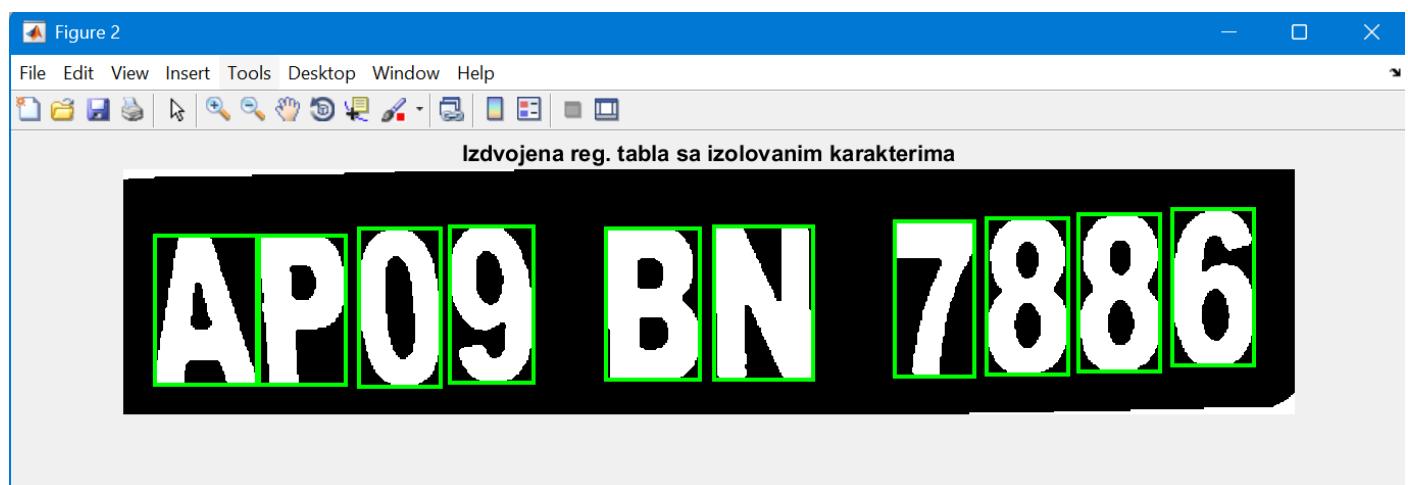
```

```

letter=readLetter(Iprops(i).Image); % Reading the letter
corresponding the binary image 'N'.
figure; imshow(Iprops(i).Image);
noPlate=[noPlate letter] % Appending every subsequent character
in noPlate variable.
end
end

```

Registracijska tabla je: APO9 BN 7886



UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

PRIMJENA NADZIRANOG I NENADZIRANOG UČENJA U MATLAB-U

1. Nadzirano učenje -> klasifikacija, regresija (binarna klasifikacija, višeklasna, algoritmi mašinskog učenja kao što su kNN, SVM, linearna regresija, logistička regresija itd.).
2. Nenadzirano učenje -> klastering.

Iris je poznati skup podataka koji se koristi u mašinskom učenju i statistici. Sadrži mjerenja cvjetova jedne biljke koja se zove Iris (to je vrsta cvijeta). Cvijet se sastoji od više dijelova, kod irisa, vanjski listovi cvijeta se zovu: **Čašični listovi** i **Laticе**. U skupu podataka nalaze se tri vrste cvijeta Iris:

1. Iris Setosa
2. Iris Versicolor
3. Iris Virginica

Vrsta cvijeta Kratki opis

Iris Setosa Manji, nježniji cvijet

Iris Versicolor Srednje veličine

Iris Virginica Najveći cvijet

Primjer1

Algoritam: Logistička regresija (binarna klasifikacija)

Logistička regresija je algoritam nadziranog učenja koji se koristi za binarne klasifikacijske probleme, odnosno kada želimo odlučiti da li neki uzorak pripada klasi 0 ili klasi 1.

Za razliku od linearne regresije, koja daje kontinuiranu numeričku vrijednost, logistička regresija daje vjerovatnoću pripadanja nekoj klasi. Zbog toga koristi sigmoid funkciju:

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Ova funkcija mapira bilo koju vrijednost u opseg (0, 1). Ako je $h(x) > 0.5 \rightarrow$ uzorak se klasificira kao klasa 1, u suprotnom kao klasa 0. Model se trenira tako što se parametri θ optimizuju tako da minimiziraju logističku (log-loss) funkciju greške:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h(x^{(i)})) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))]$$

Za optimizaciju se najčešće koristi:

- Gradijentni spust (ZA VELIKI SKUP PODATAKA)

Gradijentni spust je iterativna metoda optimizacije koja se zasniva na pomjeranju parametara u smjeru negativnog gradijenta funkcije greške. Drugim riječima, algoritam traži smjer u kojem greška najbrže opada i pomjera parametre korak po korak: $\theta := \theta - \alpha \nabla J(\theta)$

Simbol	Značenje
$\nabla J(\theta)$	gradijent funkcije greške
α	stopa učenja (learning rate), koja određuje veličinu koraka

- Newton-Raphson metoda (MALI I SREDNJI SKUP PODATAKA)

Newton-Raphson je brža optimizaciona metoda koja koristi drugi izvod (Hessovu matricu) kako bi izračunala direktniji korak prema minimumu. Ažuriranje parametara se vrši formulom:

$$\theta := \theta - H^{-1} \nabla J(\theta)$$

Simbol	Značenje
H	Hessova matrica (matrica drugih izvoda), daje zakrivljenost funkcije greške
$\nabla J(\theta)$	gradijent funkcije greške

Koristiti Iris skup podataka sa dvije klase cvijeta, Iris Setosa i Iris Versicolor, za ulazne podatke uzeti dužinu i širinu čašice. Primijeniti logističku regresiju i naučiti model da klasificuje cvijet u jednu od te dvije klase. Ako je $h(x) > 0.5$, klasa 1, inače je klasa 0.

```
% 1) Binarna klasifikacija - Logistička regresija
load fisheriris
X = meas(1:100,1:2); % biramo 100 primjera (2 klase)
y = species(1:100);

% Pretvaranje u binarnu oznaku (1 = setosa, 0 = versicolor)
y = strcmp(y, 'setosa');

% Dodavanje kolone jedinica (bias)
Xb = [ones(size(X,1),1) X];

% Inicijalizacija parametara
theta = zeros(3,1);

% Trening Newton-Raphson metodom
for i = 1:50
    z = Xb*theta;
    h = 1./(1+exp(-z));
    grad = Xb'* (h-y)/length(y);
    H = Xb' * diag(h.* (1-h)) * Xb / length(y);
    theta = theta - H\grad;
end

% Test i tačnost
y_pred = h > 0.5;
accuracy = mean(y_pred == y)*100;
disp(['Tačnost logističke regresije: ' num2str(accuracy) '%'])
```

Command Window

```
>> primjer1
Tačnost logističke regresije: 100%
```

Primjer2

k-Nearest Neighbors (kNN) je algoritam nadziranog učenja koji se koristi za klasifikaciju i regresiju, ali se najčešće primjenjuje u klasifikaciji. Osnovna ideja algoritma je da uzorak pripada istoj klasi kao i njegovi najbliži susjedi u prostoru osobina (feature space).

Princip rada

Kada se pojavi novi uzorak koji treba klasifikovati:

1. Izračunaju se udaljenosti između tog uzorka i svih uzoraka u trening skupu.
2. Odabere se k najbližih uzoraka (susjeda).
3. Gleda se kojoj klasi pripada većina tih susjeda.
4. Uzorak se klasificuje u tu najčešću klasu.

$$d(x, x_i) = \sqrt{\sum_{j=1}^n (x_j - x_{ij})^2}$$

Najčešće se koristi Euklidska udaljenost:

gdje su:

- x novi uzorak,
- x_i uzorci iz trening skupa,
- n broj osobina.

Parametar k

Broj susjeda k jako utiče na tačnost algoritma:

Vrijednost k	Posljedica
Premali k (npr. k=1)	Model postaje osjetljiv na šum, može preučno učiti (overfitting).
Prevelik k	Model previše „izravnava“ podatke, može pogrešno klasifikovati (underfitting).

Zato se kobično bira eksperimentalno (npr. k = 3, 5, 7).

Koristiti sve tri klase Iris cvijeta i primijeniti kNN algoritam.

Algoritam kNN traži najbliže primjere u trening skupu i gleda kojoj klasi većina pripada.

```
% 2) Višeklasna klasifikacija - kNN
load fisheriris
X = meas(:,1:2);
y = categorical(species); % << Ovdje dodano

% Podjela na train / test skup
idx = randperm(length(y));
train = idx(1:100);
test = idx(101:end);

Xtrain = X(train,:);
ytrain = y(train);
```

```

Xtest = X(test,:) ;
ytest = y(test) ;

k = 5;
y_pred = categorical(strings(length(ytest),1)); % pravimo prazan categorical vektor

for i = 1:length(Xtest)
    dist = sum((Xtrain - Xtest(i,:)).^2,2);
    [~, ind] = sort(dist);
    y_pred(i) = mode(ytrain(ind(1:k))); % sada radi jer su kategorije
end

accuracy = mean(y_pred == ytest)*100;
disp(['kNN tačnost: ' num2str(accuracy) '%'])

```

>> primjer2
kNN tačnost: 76%

Primjer3

Linearna regresija je algoritam nadziranog učenja koji se koristi za rješavanje regresijskih problema, odnosno za predviđanje kontinuirane numeričke vrijednosti. Cilj algoritma je da pronađe linearu vezu između ulazne varijable (nezavisne) i izlazne varijable (zavisne).

Najjednostavniji model linearne regresije može se zapisati kao:

$$y = \theta_0 + \theta_1 x$$

gdje je:

Oznaka	Značenje
y	izlaz (predviđena vrijednost)
x	ulaz (osobina / feature)
θ_0	slobodni član (intercept)
θ_1	koeficijent nagiba (weight)

Koeficijent θ_1 određuje kako se izlaz mijenja kada se ulaz promijeni.

Cilj algoritma

Pronaći parametre θ_0 i θ_1 tako da se linija najbolje prilagodi podacima.

Za to se koristi funkcija greške, najčešće Srednja kvadratna greška (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

gdje je:

- $h_\theta(x)$ izlaz modela,
- m broj podataka.

Najniža vrijednost ove funkcije znači najbolje moguće uklapanje linije sa podacima.

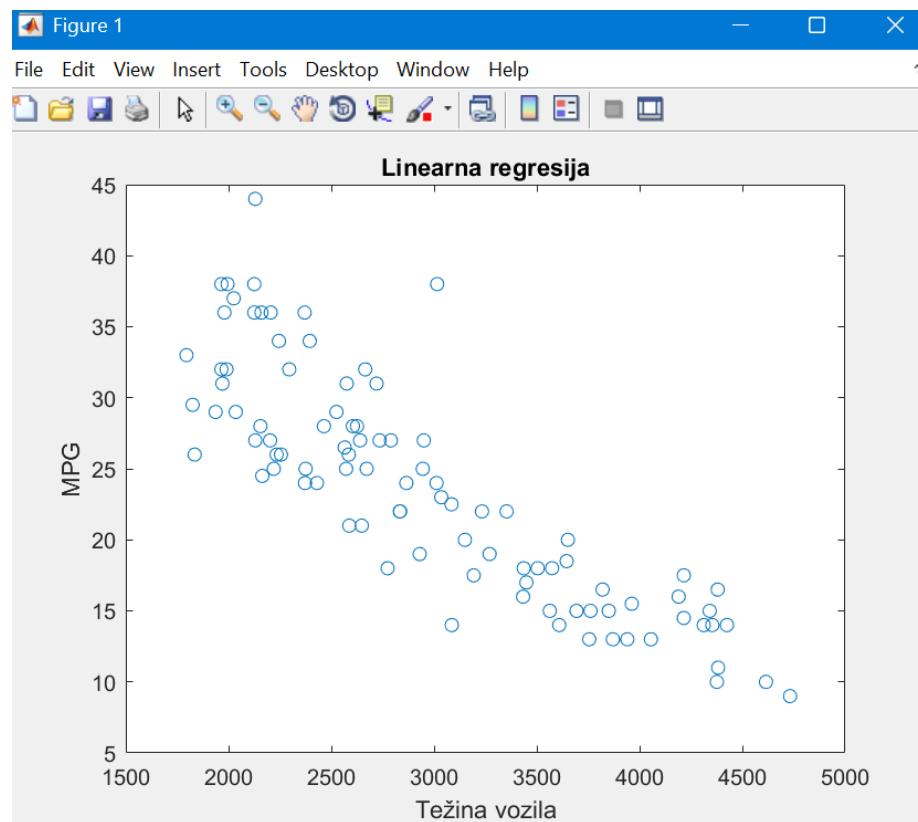
Koristiti carsmall skup podataka za modelovanje zavisnosti potrošnje goriva od težine vozila.

```
% 3) Linearna regresija
load carsmall
X = Weight;
y = MPG;

Xb = [ones(size(X)) X];
theta = (Xb' * Xb) \ (Xb' * y);

y_pred = Xb * theta;

figure
plot(X, y, 'o'); hold on
plot(X, y_pred);
xlabel('Težina vozila')
ylabel('MPG')
title('Linearna regresija')
```



Primjer4 – nenadzirano učenje

Poređenje metoda nenadziranog učenja

Metoda	Osnovna ideja	Oblik klastera	Pripadnost	Gdje se koristi
<i>k-means</i>	Centar = aritmetička sredina	Okrugli	Jedna klasa po tački	jednostavne grupe, brza analiza
<i>GMM</i>	Svaki klaster = Gaussova distribucija	Eliptični	Soft (vjerojatnoća)	statistička analiza, signali
<i>Fuzzy c-means</i>	Stepen pripadnosti svakom klasteru	Okrugli / eliptični	Fuzzy pripadnost	inteligentni sistemi, upravljanje energijom

K-means je algoritam nenadziranog učenja koji se koristi za klasterovanje podataka. Osnovna ideja je grupisati podatke u K grupa (klastera) tako da podaci unutar istog klastera budu što sličniji, a različiti klasteri budu što različitiji. Nema labela i nema unaprijed označenih klasa jer algoritam treba sam da otkriva strukturu u podacima.

Algoritam automatski traži centre grupa (tzv. centroidi) tako da minimizira varijansu unutar klastera. Svaka tačka se pridružuje onom centru kojem je najbliža.

Koraci algoritma K-means

1. Odaberite broj klastera K (npr. K = 3).
 2. Nasumično inicijalizujte K centra (centroide).
 3. Dodjeljivanje tačaka klasterima:
Za svaki podatak izračunajte udaljenost do svakog centroida i dodjelite ga klasteru čiji je centroid najbljiži.
 4. Ažuriranje centroida:
Za svaki klaster izračunajte novi centroid kao srednju vrijednost svih tačaka u tom klasteru.
 5. Provjera konvergencije:
Ako se centri više ne mijenjaju (ili promjena je vrlo mala), algoritam se zaustavlja.
Inače, vratite se na korak 3.
-

Matematička formulacija

Centroid klastera C_j izračunava se kao:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

Cilj je minimizirati funkciju greške (sumu kvadratnih udaljenosti):

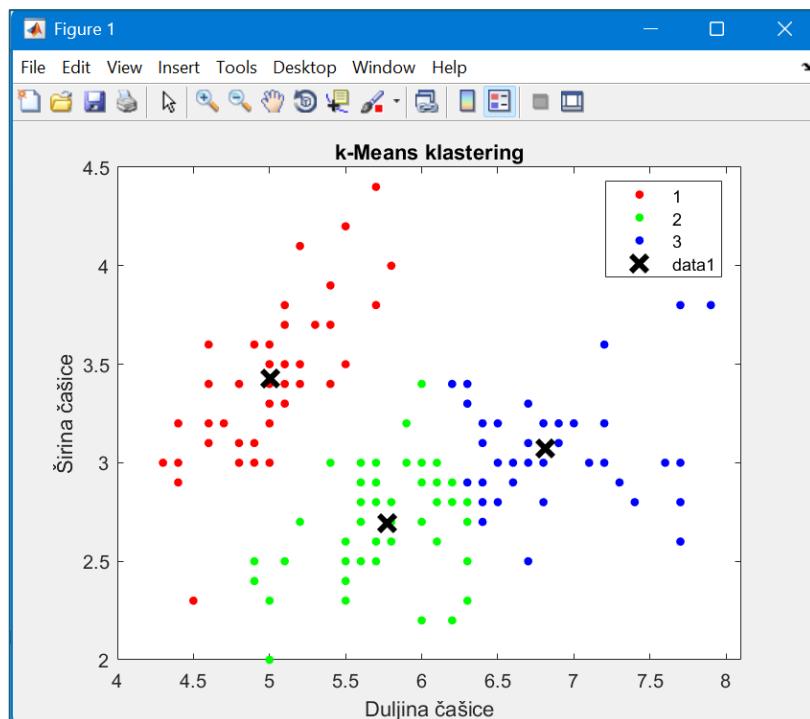
$$J = \sum_{j=1}^K \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

Koristiti k-means za automatski grupisanje cvijeća u tri grupe.

```
% 4) K-Means klastering
load fisheriris
X = meas(:,1:2);

[idx, C] = kmeans(X, 3);

figure
gscatter(X(:,1), X(:,2), idx)
hold on
plot(C(:,1), C(:,2), 'kx',
'MarkerSize',12,'LineWidth',3)
xlabel('Duljina čašice')
ylabel('Širina čašice')
title('k-Means klastering')
```



Primjer 5

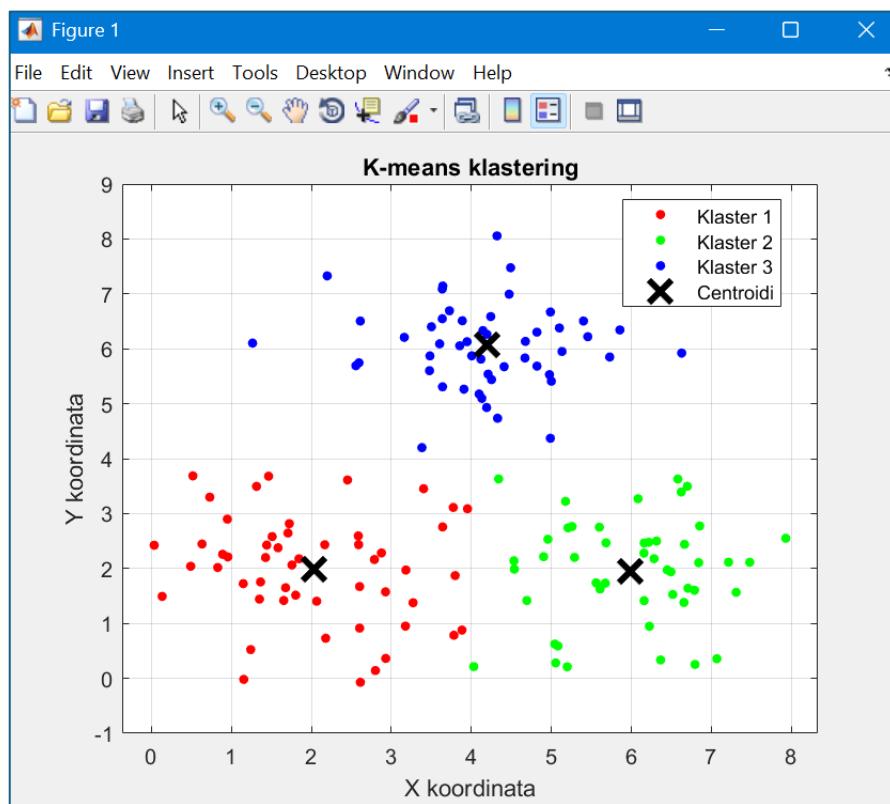
```
% K-means klastering primjer

% 1) Generišemo primjer podataka (3 različite grupe)
rng(1) % zbog ponovljivosti
group1 = randn(50,2) + [2 2];
group2 = randn(50,2) + [6 2];
group3 = randn(50,2) + [4 6];

% Spajamo sve dijelove u jednu matricu podataka
X = [group1; group2; group3];
```

```
% 2) Primjena k-means algoritma
k = 3; % broj klastera
[idx, centroids] = kmeans(X, k);

% 3) Vizualizacija rezultata
figure;
gscatter(X(:,1), X(:,2), idx, 'rgb');
hold on;
plot(centroids(:,1), centroids(:,2), 'kx', 'MarkerSize', 15, 'LineWidth', 3);
title('K-means klastering');
xlabel('X koordinata');
ylabel('Y koordinata');
legend('Klaster 1', 'Klaster 2', 'Klaster 3', 'Centroidi');
grid on;
```



ELBOW METODA – metoda za određivanje optimalnog broja klastera

K-means zahtijeva da unaprijed zadamo broj klastera **K**. Elbow metoda pomaže da odaberemo najbolji K. Mjeri se ukupna suma kvadrata udaljenosti tačaka od njihovih centara (tzv. *Within-Cluster Sum of Squares – WCSS*) za različite vrijednosti K.

- Ako je K premali → grupe su “loše”, WCSS je velik.
- Kako se K povećava → WCSS opada.
- Ali nakon jednog trenutka smanjenje postane zanemarivo → tu je “lakat” (elbow).

Primjer 6

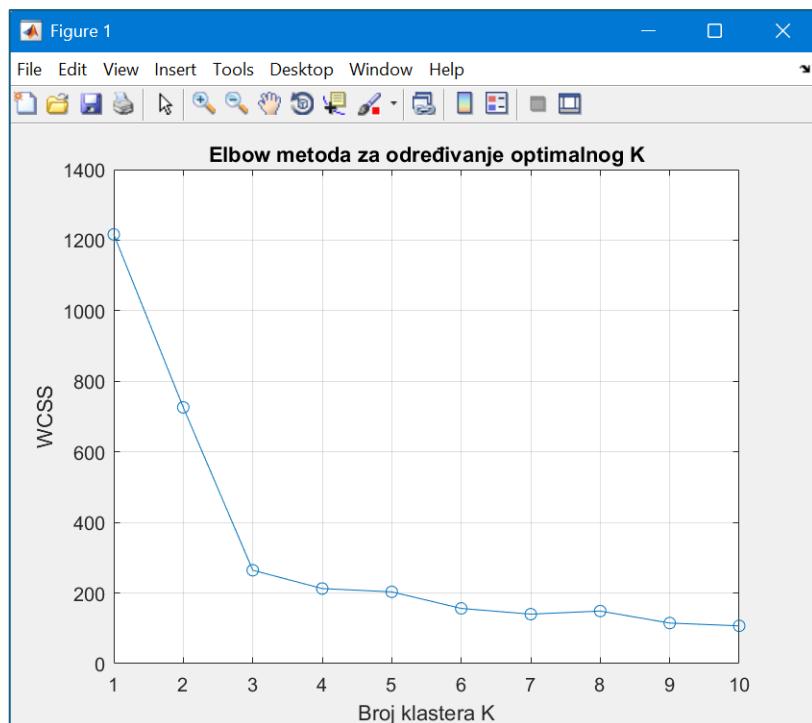
```
% Elbow metoda za određivanje optimalnog K

rng(1)
X = [randn(50,2)+[2 2];
      randn(50,2)+[6 2];
      randn(50,2)+[4 6]];

maxK = 10;
WCSS = zeros(maxK,1);

for k = 1:maxK
    [~,~,sumd] = kmeans(X, k);
    WCSS(k) = sum(sumd);
end

figure;
plot(1:maxK, WCSS, '-o');
xlabel('Broj klastera K');
ylabel('WCSS');
title('Elbow metoda za određivanje optimalnog K');
grid on;
```



K = 3

GAUSS klastering (GMM – Gaussian Mixture Model)

Ovdje klasteri nisu bazirani na sredini (kao k-Means), već se svaki klaster modelira kao Gaussova distribucija. Koristi se EM algoritam (Expectation-Maximization).

Primjer 7

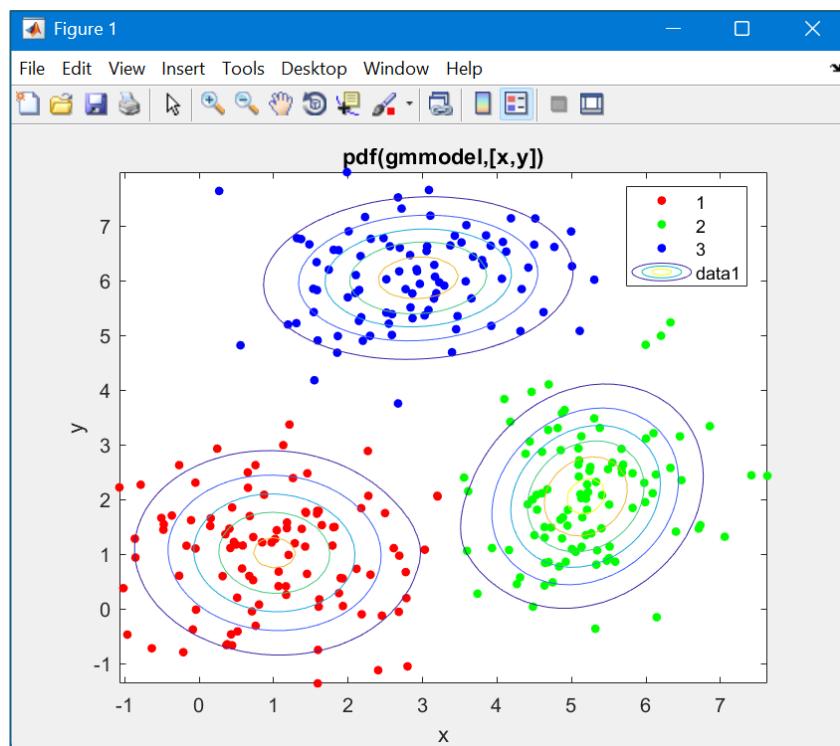
```
% Gaussian Mixture Model klastering

rng(1)
X = [randn(100,2)+[1 1];
      randn(100,2)+[5 2];
      randn(100,2)+[3 6]];

gmmmodel = fitgmdist(X, 3);      % 3 klastera
idx = cluster(gmmmodel, X);    % dodjela tačaka klasterima

figure;
gscatter(X(:,1), X(:,2), idx);
title('GMM klasterovanje (Gaussian Mixture Model)');
hold on;

% crtanje kontura
f = @(x,y) pdf(gmmmodel,[x y]);
ezcontour(f, [min(X(:,1)) max(X(:,1))] , [min(X(:,2))
max(X(:,2))]);
```



Fuzzy c-means (Fuzzy Logika klastering)

Za razliku od k-means gdje svaka tačka pripada samo jednom klasteru, u fuzzy c-means svaka tačka ima stepen pripadnosti svakom klasteru (npr. 0.2, 0.7, 0.1). Koristi se kod inteligentnih sistema, neuronskih mreža, upravljanja energijom...

Primjer 8

```
% Fuzzy C-Means klastering

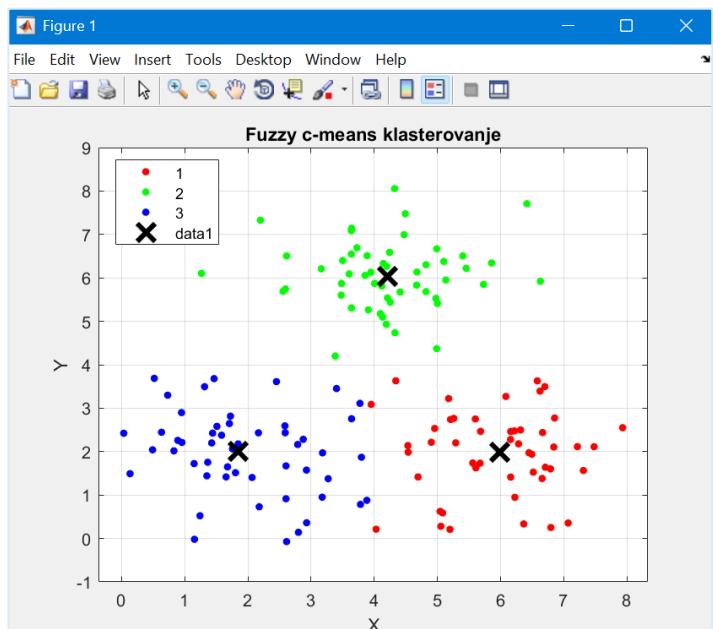
rng(1)
X = [randn(50,2)+[2 2];
      randn(50,2)+[6 2];
      randn(50,2)+[4 6]];

% fuzzy c-means
[center, U] = fcm(X, 3); % U = matrica pripadnosti

% Dodjela klastera prema najvećoj pripadnosti
[~, idx] = max(U);

figure;
gscatter(X(:,1), X(:,2), idx);
hold on;
plot(center(:,1), center(:,2), 'kx', 'MarkerSize', 15,
'LineWidth', 3);
title('Fuzzy c-means klasterovanje');
xlabel('X'); ylabel('Y');
grid on;
```

```
>> primjer8
Iteration count = 1, obj. fcn = 518.699270
Iteration count = 2, obj. fcn = 396.905090
Iteration count = 3, obj. fcn = 376.562145
Iteration count = 4, obj. fcn = 331.826143
Iteration count = 5, obj. fcn = 254.244822
Iteration count = 6, obj. fcn = 201.824826
Iteration count = 7, obj. fcn = 195.237961
Iteration count = 8, obj. fcn = 194.789717
Iteration count = 9, obj. fcn = 194.755255
Iteration count = 10, obj. fcn = 194.751797
Iteration count = 11, obj. fcn = 194.751360
Iteration count = 12, obj. fcn = 194.751295
Iteration count = 13, obj. fcn = 194.751284
Iteration count = 14, obj. fcn = 194.751282
```



UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

NEURONSKE MREŽE ZA DEEP LEARNING

1) Feed-forward neural network (FNN fully-connected) je osnovni tip umjetne neuronske mreže, poznat i kao višeslojni perceptron (MLP Multi-Layer Perceptron). Njena glavna karakteristika je da informacija teče samo u jednom smjeru, od ulaznog sloja, kroz jedan ili više skrivenih slojeva, prema izlaznom sloju, bez povratnih veza (nema ciklusa). Kod višeslojne mreže, izlaz svakog sloja postaje ulaz narednom sloju. Mreža se sastoji od:

1. Ulaznog sloja – prima podatke (feature vektori).
2. Skrivenih slojeva – svaki neuron računa ponderisanu sumu ulaza, dodaje bias i prolazi kroz nelinearnu aktivacijsku funkciju (npr. sigmoid, tanh, ReLU).
3. Izlaznog sloja – daje predikciju (klasifikaciju ili vrijednost regresije).

Matematički, izlaz jednog neurona se računa kao: $y = f(Wx + b)$

gdje je:

W – matrica težina,

x – vektor ulaza,

b – bias (pomak),

$f(\cdot)$ – nelinearna aktivacijska funkcija.

Treniranje FNN mreže se zasniva na algoritmu *propagacije greške unazad (backpropagation)* i **optimizaciji težina** pomoću metoda kao što su:

- Levenberg–Marquardt (trainlm) – vrlo brza konvergencija za male do srednje mreže.
- Scaled conjugate gradient (trainscg) – efikasna za veće mreže i veće skupove podataka.
- Gradient descent (traingd) – jednostavnija, ali sporija metoda.

Tok treniranja:

1. Propagacija ulaza naprijed → izračun trenutnog izlaza.
2. Izračun greške:

$$e = y_{target} - y_{predicted}$$

3. Propagacija greške unazad i ažuriranje težina:

$$W_{new} = W_{old} + \Delta W$$

gdje se ΔW računa na osnovu gradijenta greške po težinama.

Proces se ponavlja kroz više epoha dok se greška ne minimizira (npr. MSE – Mean Squared Error).

Feed-forward mreže su univerzalni aproksimatori i koriste se u širokom spektru problema:

- Regresija (predviđanje numeričkih vrijednosti),
- Klasifikacija (prepoznavanje uzoraka, signala, slike),
- Upravljanje i automatizacija (npr. aproksimacija nelinearnih sistema),
- Predviđanje vremenskih serija (kada se koristi uz “lagged inputs”).

Primjer1

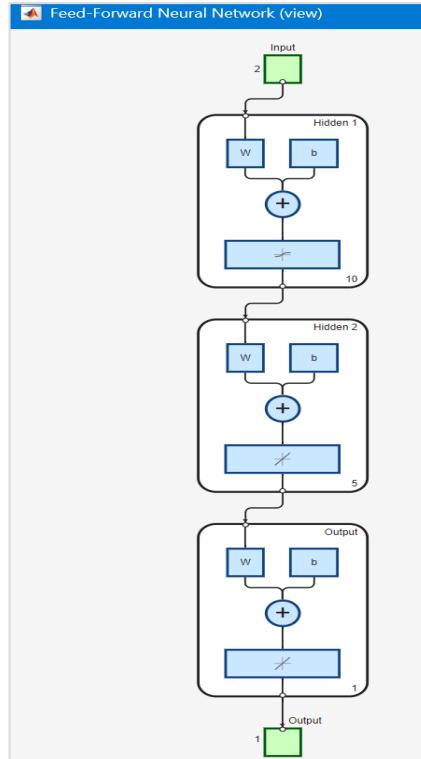
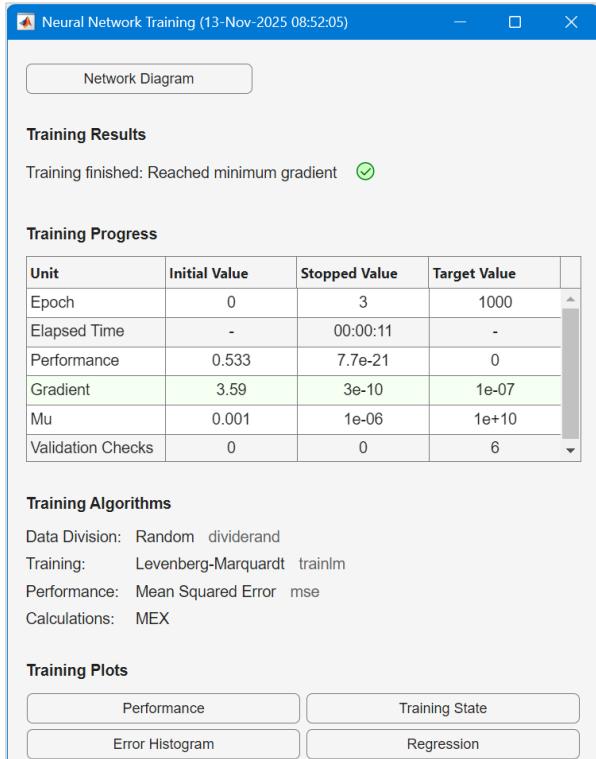
```
% Ulazni i cilj podaci
X = [0 0 1 1; 0 1 0 1]; % 2 ulaza, 4 uzorka
T = [0 1 1 0]; % ciljna vrijednost (XOR problem)

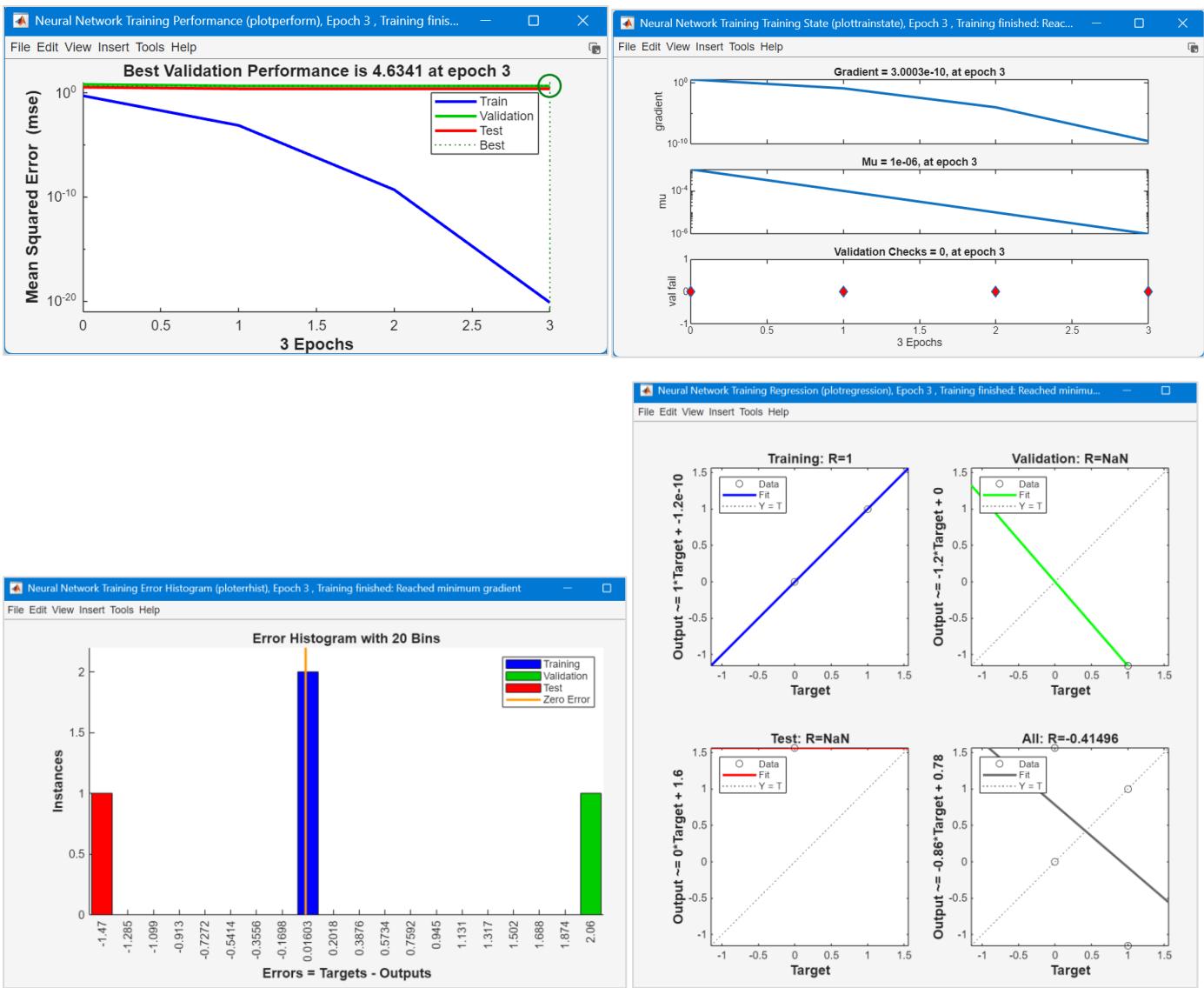
% Kreiranje mreže (2 ulaza, 2 skrivena sloja)
net = feedforwardnet([10 5]);
net.trainFcn = 'trainlm'; % algoritam učenja (Levenberg-Marquardt)
net.layers{1}.transferFcn = 'tansig'; % aktivacijska funkcija 1. sloja
net.layers{2}.transferFcn = 'purelin'; % izlazni sloj (linearan)

% Treniranje
net = train(net, X, T);

% Testiranje
Ypred = net(X);
perf = perform(net, T, Ypred);
disp(['Greška mreže (MSE): ', num2str(perf)]);
```

```
>> primjer1
Greška mreže (MSE): 1.7695
```





Primjer2 Primjer koristenjem Deep Learning Toolbox-a

```
% Generisanje podataka (regresija)
X = linspace(-2*pi, 2*pi, 200)';
T = sin(X) + 0.1*randn(size(X));

% Definicija slojeva FNN mreže
layers = [
    featureInputLayer(1)
    fullyConnectedLayer(10)
    reluLayer
    fullyConnectedLayer(5)
    reluLayer
    fullyConnectedLayer(1)
    regressionLayer
];

% Kreiranje layerGraph
lgraph = layerGraph(layers);
```

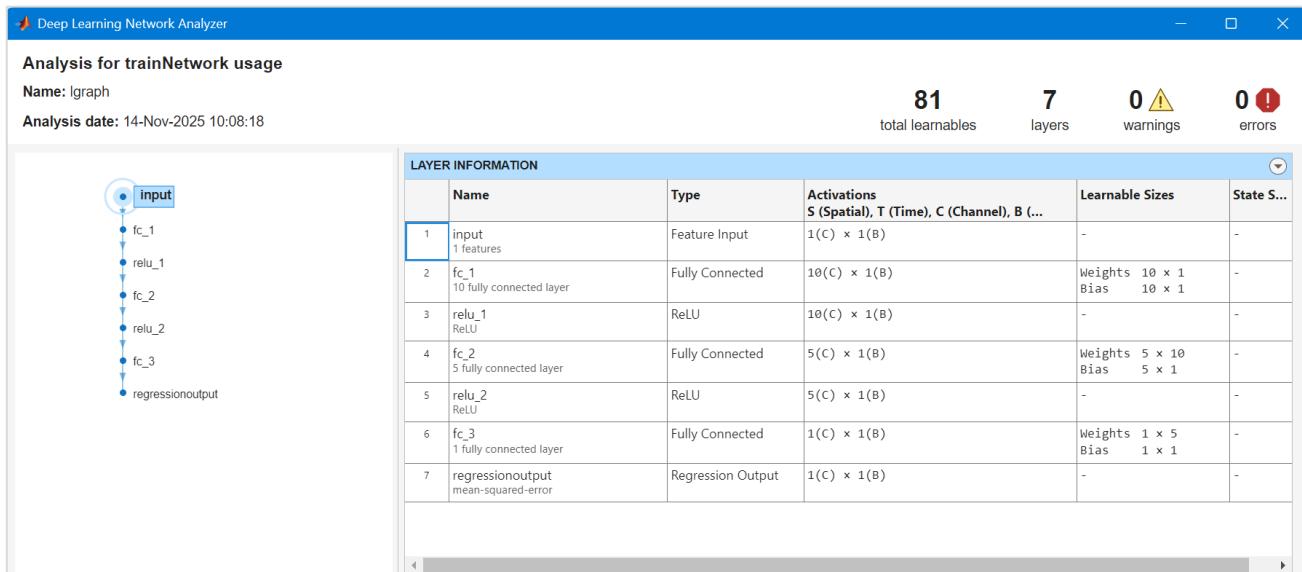
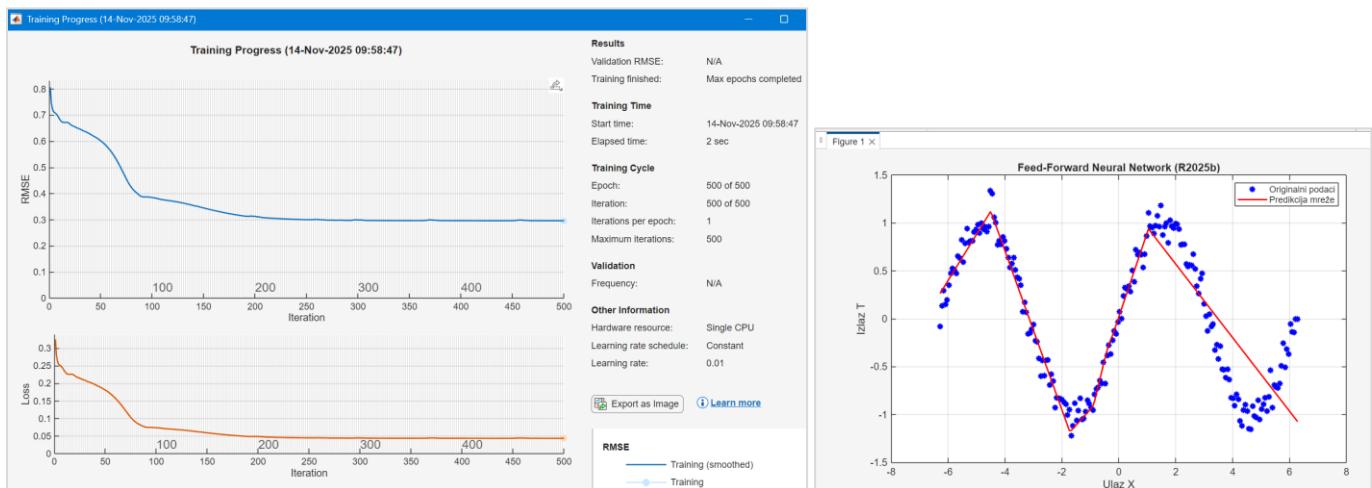
```
% Analiza mreže - direktno prikazati arhitekturu mreže naredbom:
analyzeNetwork(lgraph);

% Opcije treniranja
options = trainingOptions('adam', ...
    'MaxEpochs', 500, ...
    'InitialLearnRate', 0.01, ...
    'Plots','training-progress', ...
    'Verbose', false);

% Treniranje
net = trainNetwork(X, T, layers, options);

% Predikcija
Ypred = predict(net, X);

figure;
plot(X, T, 'b*', X, Ypred, 'r', 'LineWidth',1.5);
legend('Originalni podaci','Predikcija mreže');
grid on;
```



2) Konvolucijska neuronska mreža - Convolutional Neural Network (CNN)

je specijalizirana duboka neuronska mreža dizajnirana da efikasno obrađuje podatke koji imaju prostornu ili vremensku strukturu, kao što su slike (2D), videosignalni (3D), vremenske serije (1D) i senzorski podaci. CNN je nastao s idejom da se neuronska mreža može naučiti prepoznati lokalne obrasce (edges, teksture, oblike), umjesto da uči direktno iz sirovih piksela kao FNN.

CNN neuronska mreža sadrži više filtra (kernela) koji klize preko slike i izvlače lokalne značajke. Filter je npr. 3×3 ili 5×5 matrica koja se konvoluirala nad ulazom. CNN konvolucije uče ivice, strukture, uglove, teksture, složenije oblike u dubljim slojevima... Učenje se zasniva na gradijentima, filteri se automatski optimiziraju. Bez nelinearnosti CNN se svodi na linearnu transformaciju, ReLU (ili drugi) aktivacijski sloj dodaje nelinearnost, npr. $\text{ReLU}(x) = \max(0, x)$

Pooling layer - Najčešće max-pooling (2×2), cilj ovog sloja je smanjenje dimenzije slike, smanjenje broja parametara, ekstrakcija dominatnih značajki (maximum pooling) i robustnost na translacije.

Fully-connected slojevi - Na kraju CNN-a nalazi se 1–3 potpuno povezana sloja koji izvode klasifikaciju i regresiju.

Output layer - softmaxLayer + classificationLayer (klasifikacija) i regressionLayer (regresija).

Postupak rada CNN mreže:

1. Ulazna slika prolazi konvolucijske filtere → detekcija lokalnih struktura
2. ReLU čisti negativne vrijednosti
3. Max-pooling smanjuje dimenziju → apstrahuje značajke
4. Više konvolucijskih blokova radi sve apstraktnije reprezentacije
5. Fully-connected slojevi vrše klasifikaciju ili regresiju
6. Mreža se trenira backpropagation algoritmom

CNN se posebno ističe jer automatski uči značajke, za razliku od tradicionalnog računarstva slike koje zahtijeva ručni feature engineering.

Primjer3

```
% Učitavanje MNIST dataseta
[XTrain, YTrain] = digitTrain4DArrayData;
[XTest, YTest] = digitTest4DArrayData;

% Definicija CNN arhitekture
layers = [
    imageInputLayer([28 28 1], 'Normalization','none') Ulazna slika 28x28x1
    (sivi nivo)

    convolution2dLayer(3, 16, 'Padding','same') Konvolucijski sloj sa 16
    filtera od 3x3 – detektuje lokalne ivice
    batchNormalizationLayer BatchNorm ubrzava treniranje i stabilizira
    gradijente
    reluLayer ReLU uvodi nelinearnost
```

```

maxPooling2dLayer(2, 'Stride',2) Max-pooling smanjuje dimenziju slike na
14x14.

convolution2dLayer(3, 32, 'Padding', 'same') Drugi konvolucijski blok uči
složenije znake
batchNormalizationLayer
reluLayer

maxPooling2dLayer(2, 'Stride',2)

convolution2dLayer(3, 64, 'Padding', 'same') Treći blok radi apstraktnije
reprezentacije (oblici cifri)
batchNormalizationLayer
reluLayer

fullyConnectedLayer(10) Fully-connected dio klasificira cifru 0-9.
softmaxLayer Softmax daje vjerovatnoću za svaku cifru.
classificationLayer
];

% Kreiranje grafičke strukture za analizu
lgraph = layerGraph(layers);

% Analiza mreže u R2025b (ispravno)
analyzeNetwork(lgraph);

% Opcije treniranja
options = trainingOptions('adam', ...
    'MaxEpochs', 5, ...
    'MiniBatchSize', 128, ...
    'Plots', 'training-progress', ...
    'Verbose', false);

% Treniranje CNN mreže
net = trainNetwork(XTrain, YTrain, layers, options);

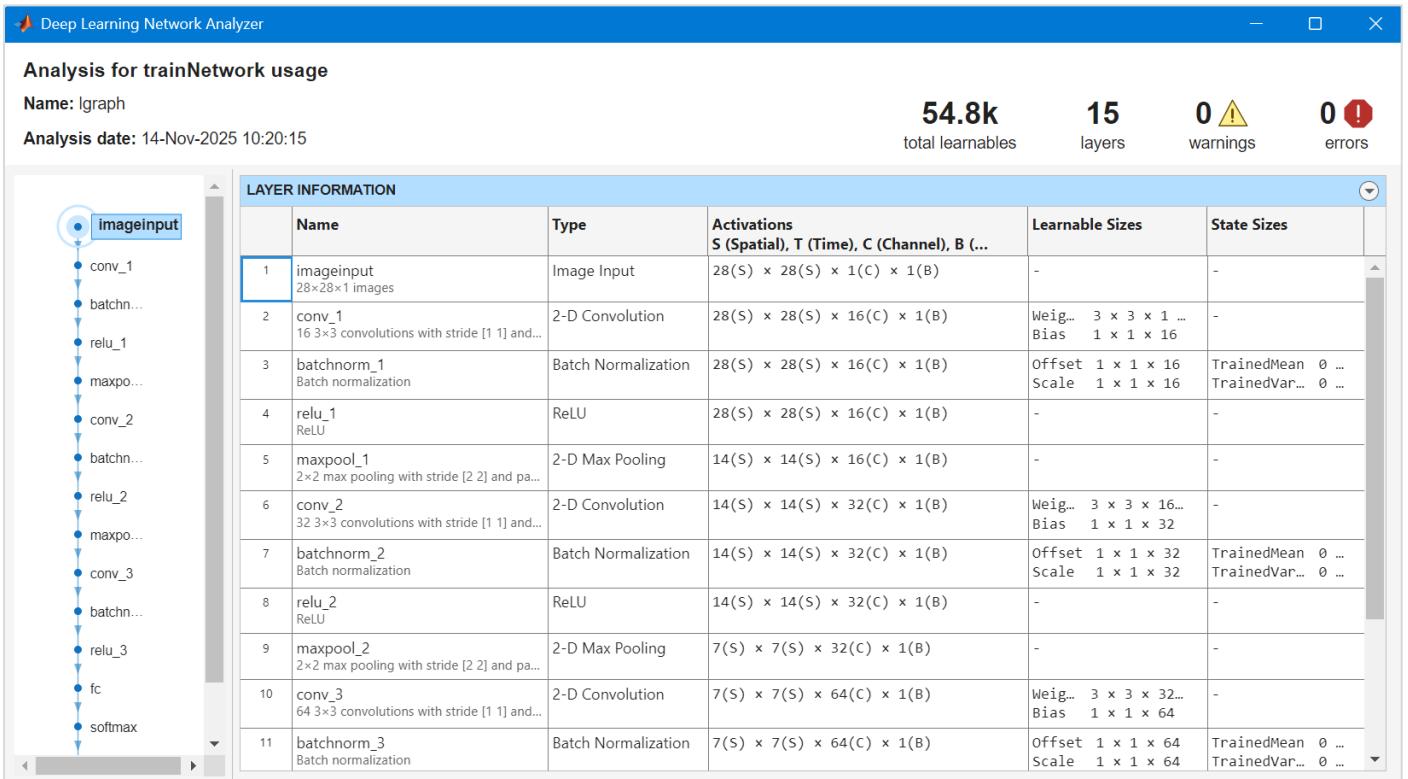
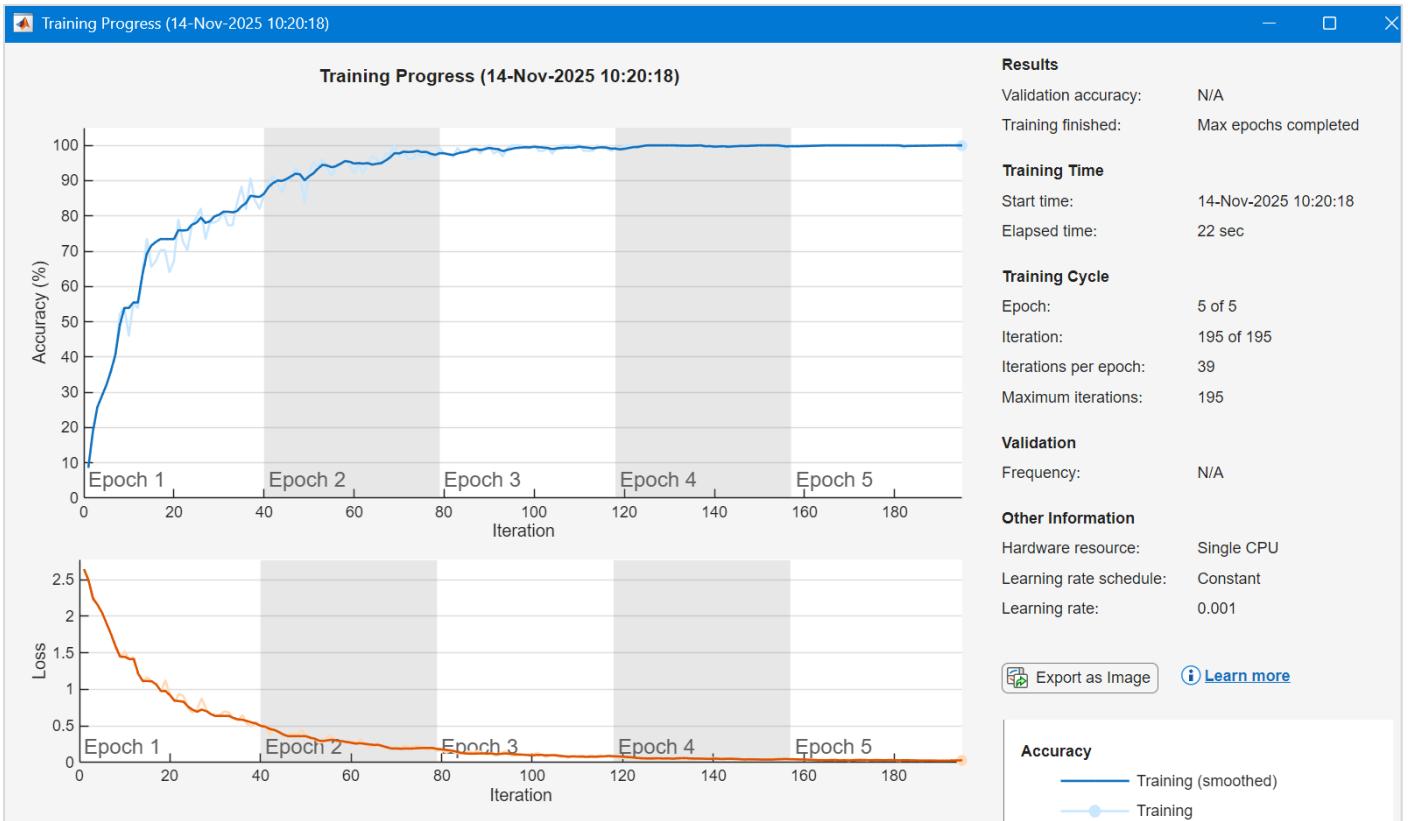
% Evaluacija
YPred = classify(net, XTest);
accuracy = mean(YPred == YTest);

disp("Tačnost CNN mreže: " + accuracy);

```

Command Window

Tačnost CNN mreže: 0.9952



3) Generativna suparnička neuronska mreža – *Generative Adversarial Neural Network (GAN)*

- sastoji se od dvije neuronske mreže koje se treniraju istovremeno, ali sa suprotnim ciljevima:
 1. *Generator (G)* – stvara nove primjere (slike, podatke) na osnovu slučajnog ulaza (latentnog prostora).
 2. *Discriminator (D)* – pokušava razlikovati stvarne podatke od lažnih koje generiše G.

Obje mreže se takmiče u igri nulte sume, tj. generator nastoji prevariti discriminator, dok discriminator pokušava prepoznati lažne podatke. Rezultat je generator koji s vremenom postaje izuzetno dobar u stvaranju podataka sličnih originalnim.

Način rada GAN neuronske mreže:

Uzimaju se stvarni podaci (npr. slike), zatim generator uzima random vektor z (latent vector) i iz njega pokušava generisati lažni primjer, discriminator dobija stvarne primjere i označi ih kao *real* i generisane primjere i označi ih kao *fake*. Nakon toga generator i diskriminator se treniraju naizmjenično, discriminator maksimalno pokušava povećati vjerovatnoću ispravne klasifikacije, dok generator minimizuje sposobnost discriminatora da otkrije lažne primjere.

Primjer4

```
%> =====
%>     GAN primjer – MATLAB R2025b
%>     Generiše 2D tačke u obliku kruga
%>     Fully functional sa dlfeval za gradient
%> =====

clear; clc; close all;

%> 1) Dataset: 2D tačke u obliku kruga (REAL DATA)
numReal = 5000;
theta = 2*pi*rand(numReal,1);
r = 1 + 0.1*randn(numReal,1);
XReal = [r.*cos(theta), r.*sin(theta)]; % 2D kružnica

%> 2) DEFINICIJA GENERATORA
latentDim = 10; % dimenzija latentnog vektora

layersG = [
    featureInputLayer(latentDim, 'Normalization', 'none')
    fullyConnectedLayer(32)
    reluLayer
    fullyConnectedLayer(16)
    reluLayer
    fullyConnectedLayer(2)           % generisana 2D tačka
];
lgraphG = layerGraph(layersG);
dlnetG = dlnetwork(lgraphG);
```

```

%% 3) DEFINICIJA DISKRIMINATORA
layersD = [
    featureInputLayer(2, 'Normalization', 'none')
    fullyConnectedLayer(32)
    leakyReluLayer(0.2)
    dropoutLayer(0.3)

    fullyConnectedLayer(16)
    leakyReluLayer(0.2)

    fullyConnectedLayer(1)
    sigmoidLayer % izlaz = vjerovatnoća real/fake
];

lgraphD = layerGraph(layersD);
dlnetD = dlnetwork(lgraphD);

%% 4) OPCIJE TRENIRANJA
numEpochs = 3000;
learningRate = 0.0005;
miniBatchSize = 64;

%% 5) TRENIRANJE GAN-A
figure;

for epoch = 1:numEpochs

    % === MINI-BATCH REALNIH PODATAKA ===
    idx = randperm(numReal, miniBatchSize);
    XBatch = XReal(idx,:);
    dlXReal = dlarray(single(XBatch'), 'CB'); % C = feature, B = batch

    % === GENERISANJE LATENT VEKTORA ===
    Z = randn(latentDim, miniBatchSize, 'single');
    dlZ = dlarray(Z, 'CB'); % C = latentDim, B = batch

    % === Izračun gradijenata sa dlfeval ===
    [gradientsD, gradientsG, lossD, lossG] = dlfeval(@modelGradients, dlnetD,
dlnetG, dlXReal, dlZ);

    % === Update mreža ===
    dlnetD = dlupdate(@(p,g) p - learningRate*g, dlnetD, gradientsD);
    dlnetG = dlupdate(@(p,g) p - learningRate*g, dlnetG, gradientsG);

    % === PRIKAZ PROGRESIJE ===
    if mod(epoch, 100) == 0
        dlXFake = forward(dlnetG, dlZ);
        fake = extractdata(dlXFake)';
        plot(XReal(:,1), XReal(:,2), 'b.');
        plot(fake(:,1), fake(:,2), 'r.');
        legend('Real', 'Fake');
        title("GAN Epoch " + epoch + ...
              " | LossD=" + gather(extractdata(lossD)) + ...
              " | LossG=" + gather(extractdata(lossG)));
    end
end

```

```

        grid on; axis equal;
        drawnow;
        hold off;
    end
end

disp('Trening GAN-a je završen.');

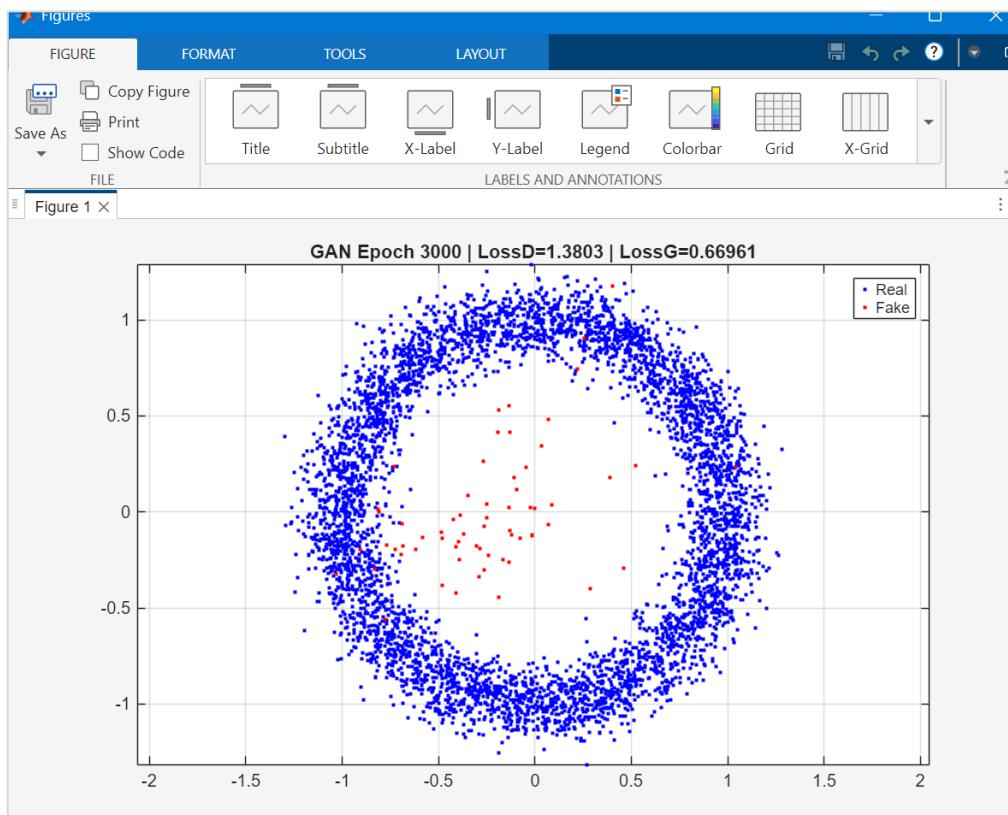
%% =====
% Funkcija za gradijente generatora i diskriminatora
%% =====
function [gradientsD, gradientsG, lossD, lossG] = modelGradients(dlnetD,
dlnetG, dlXReal, dlZ)
    % Forward pass generatora
    dlXFake = forward(dlnetG, dlZ);

    % Forward pass discriminatora
    dReal = forward(dlnetD, dlXReal);
    dFake = forward(dlnetD, dlXFake);

    % Loss funkcije
    lossD = -mean(log(dReal + 1e-8) + log(1 - dFake + 1e-8));
    lossG = -mean(log(dFake + 1e-8));

    % Gradijenti
    gradientsD = dlgradient(lossD, dlnetD.Learnables);
    gradientsG = dlgradient(lossG, dlnetG.Learnables);
end

```



4) Rekurentna neuronska mreža (RNN) – Recurrent Neural Network (RNN)

- je vrsta neuronske mreže dizajnirana za obradu sekvenci podataka, vremenskih nizova, govora, signalnih serija, tekstova itd. RNN ima memoriju. Svaki izlaz zavisi od trenutnog ulaza i od stanja iz prethodnog vremenskog koraka (hidden state). Matematički izraz za RNN:

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$
$$y_t = W_y h_t + c$$

Ova povratna petlja daje mreži sposobnost da:

- uči vremenske obrasce
- modelira dinamiku sistema
- radi predikciju sekvenci

Klasična RNN predstavlja osnovni model za obradu sekvencijalnih podataka, ali posjeduje značajna ograničenja kada se radi o učenju dugoročnih zavisnosti unutar vremenskih serija. Najveći izazov je takozvani problem nestajućeg gradijenta (vanishing gradient). Tokom treniranja, RNN propagira grešku kroz veliki broj vremenskih koraka. Zbog višestrukog množenja malih vrijednosti gradijenta, informacija o udaljenim događajima u sekvenci se brzo gubi, pa mreža postaje nesposobna da nauči dugoročne obrasce. Ovo rezultira:

- slabom memorijom mreže,
- nemogućnošću učenja dugih sekvenci,
- otežanim treniranjem i sporom konvergencijom.

Da bi se riješio ovaj problem, razvijene su naprednije arhitekture LSTM (Long Short-Term Memory) i GRU (Gated Recurrent Unit), koje koriste unutrašnje kontrolne mehanizme – „vrata“ (gates) – za regulaciju protoka informacija kroz vrijeme.

LSTM mreža uvodi strukture nazvane *ćelijsko stanje* i tri vrste vrata (input, forget i output gates). Ova vrata omogućavaju mreži da selektivno pamti ili zaboravlja informacije kroz duge vremenske intervale. Zahvaljujući tome, LSTM značajno ublažava problem nestajućeg gradijenta i omogućava učenje kompleksnih dugoročnih zavisnosti u sekvencama.

GRU je pojednostavljena verzija LSTM arhitekture, ali sa sličnim performansama. Koristi dva vrata (reset i update gate) i nema posebno ćelijsko stanje, što je čini bržom i računski efikasnijom u odnosu na LSTM. GRU daje odlične rezultate u sistemima gdje je potreban kompromis između tačnosti i brzine treniranja.

Primjer5

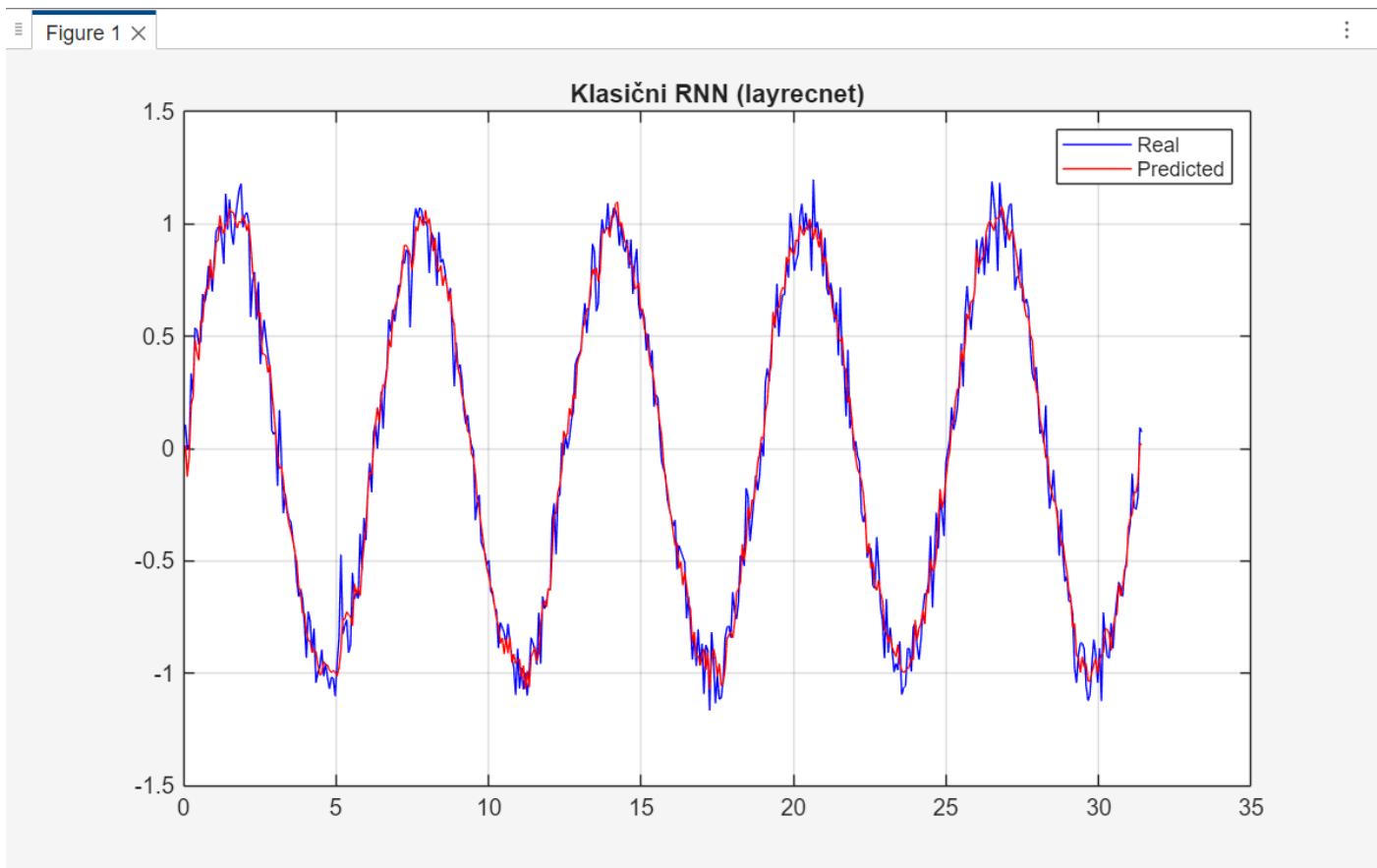
```
clear;
clc;
close all;

t = linspace(0,10*pi,500);
Y = sin(t)+0.1*randn(size(t));
```

```
% layrecnet očekuje sekvence u cell-u
Xseq = num2cell(Y(1:end-1));
Tseq = num2cell(Y(2:end));

% Kreiraj rekurentnu mrežu (jedan skriveni sloj, 20 neurona)
net = layrecnet(1:1,20); % delay 1, 20 neurona
net.trainFcn = 'trainlm';
net.divideFcn = 'divideTrain';
net = train(net, Xseq, Tseq);
Ypred_seq = net(Xseq);
Ypred = cell2mat(Ypred_seq);

figure;
plot(t(2:end), Y(2:end), 'b'); hold on;
plot(t(2:end), Ypred, 'r');
legend('Real', 'Predicted');
title('Klasični RNN (layrecnet)');
grid on;
```



5) Transformeri

Transformeri su arhitektura zasnovana na mehanizmu self-attention koja zamjenjuje rekurentne i konvolucijske pristupe u zadacima sekvenci (NLP) i, adaptacijom (ViT), u računalnom vidu. Osnovna ideja: model uči odnose *svakog* elementa sekvence sa *svim* ostalim elementima kroz težinske pozorne mehanizme, paralelizirano.

NLP (Natural Language Processing) *Obrada prirodnog jezika* je oblast umjetne inteligencije koja se bavi razumijevanjem i generisanjem ljudskog jezika: analiza teksta, prevodenje, klasifikacija emocija, chatbotovi, sažimanje teksta, detekcija namjere, entiteta itd.

ViT (Vision Transformer) *Vizijski transformer* je arhitektura transformera prilagođena za obradu slika. Umjesto riječi (tokena), slika se dijeli na male blokove, *patch-e*. Svaki patch se tretira kao token u transformer arhitekturi. ViT se koristi za: klasifikaciju slika, detekciju objekata, segmentaciju, generisanje slika itd.

Ključni elementi transformesa:

- Self-attention (scaled dot-product) Za ulazne reprezentacije X transformer računa upite (Q), ključeve (K) i vrijednosti (V):

$$Q = XW_Q, K = XW_K, V = XW_V$$

Attention izračun:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

gdje d_k je dimenzija ključeva.

- Multi-head attention: paralelno izvršavanje više self-attention "glava", koje se potom spoje i projektuju — omogućava modelu da uči različite vrste odnosa.
- Feed-forward block: nakon attention bloka ide pozicioni-nezavisani MLP (obično dva FC sloja sa ReLU), sa residual vezama i layer normalization.
- Positional encoding: budući da attention ne nosi informaciju o poretku elemenata, dodaju se položajne kodove (sinusoidalne ili učene) $PE(pos)$ na ulazne tokene. (MathWorks ima `sinusoidalPositionEncodingLayer`)

UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

AGENTI ~ u vještačkoj inteligenciji AI Agents

Agent za vještačku inteligenciju je autonomni sistem koji prikuplja informacije iz okruženja putem senzora, obrađuje podatke pomoću internog modela, donosi odluke pomoću algoritama učenja ili optimizacije i/ili djeluje nad okruženjem putem aktuatora. Opisuje se funkcijom:

$$f: S \rightarrow A$$

Gdje je S skup stanja okruženja, a A skup svih mogućih rješenja.

Razlikujemo nekoliko vrsta AI agenata:

1. Reaktivni agenti (Simple Reflex Agents)

Reaktivni agenti donose odluke isključivo na osnovu trenutnog stanja okruženja, bez korištenja memorije prošlih događaja. Reakcije se baziraju na pravilo–akcija (if–then) strukturama.

$$a = f(s)$$

Gdje je s trenutno stanje, a a akcija.

Karakteristike ovog agenta su da ne pamte prethodna stanja, jednostavnii su i brzi, pa su zbog toga pogodni za deterministička okruženja.

Primjer1

Reaktivni agent - primjer regulacije napona

```
Vmeas = 10.5; % izmjereni napon
Vref  = 12;

if Vmeas < Vref
    action = 'increase_duty';
else
    action = 'decrease_duty';
end

disp(action)
```

Command Window

```
Warning: Unable to create personal
>> primjer1
increase_duty
```

2. Model-bazirani agenti

Model-bazirani agenti posjeduju memoriju stanja i interni matematički ili logički model okruženja. Oni ne donose odluke samo na osnovu trenutnog stanja, već i koriste historiju sistema. Opisuje se funkcijom:

$$s_t^{int} = f(s_t, s_{t-1}, \dots, s_0)$$

Model-bazirani agenti imaju bolje ponašanje u dinamičkim sistemima i bolju sposobnost predviđanja.

Primjer2

model_based_agent.m

```
function action = model_based_agent(Vmeas)
    % Model-bazirani agent sa memorijom

    persistent prevV

    if isempty(prevV)
        prevV = 0;
    end

    if Vmeas < prevV
        action = 'increase_duty';
    else
        action = 'decrease_duty';
    end

    prevV = Vmeas;
end
```

```
clc; clear;

a1 = model_based_agent(10.2)
a2 = model_based_agent(10.0)
a3 = model_based_agent(11.5)
```

The screenshot shows the MATLAB Command Window with the title "Command Window". It displays the following code execution:

```
a1 =
'decrease_duty'

a2 =
'increase_duty'

a3 =
'decrease_duty'
```

The output shows three assignments: a1 is set to 'decrease_duty', a2 is set to 'increase_duty', and a3 is set to 'decrease_duty'.

3. Ciljno-orientisani agenti

Ciljno orijentisani agenti djeluju na osnovu jasno definisanog cilja i koriste algoritme planiranja da bi odredili niz koraka koji vode ka cilju. Za razliku od reaktivnih agenata ovi ne reaguju samo na stanje već razmatraju buduća stanja.

Opisuje se funkcijom:

$$A = \{a_1, a_2, \dots, a_n\}$$

Primjer 3

```
% na primjer cilj je dostizanje referentnog napona

Vtarget = 12;
V = 9;

while abs(V - Vtarget) > 0.1
    if V < Vtarget
        V = V + 0.5;
    else
        V = V - 0.5;
    end
end

disp("Cilj postignut: " + V)
```

```
>> primjer3
Cilj postignut: 12
```

4. Utility-based agenti

Utility-based agenti ne biraju akcije samo da bi postigli cilj, već da bi maksimizirali funkciju korisnosti (utility function). Ovi agenti su pogodni za optimizaciju.

Funkcija korisnosti:

$$U(s) = \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n$$

Primjer4

```
V = 11.7;
I = 3.2;

w1 = 0.7;
w2 = 0.3;

U = w1*(1/abs(12-V)) + w2*I;

disp("Utility vrijednost: " + U)
```

```
>> primjer4
Utility vrijednost: 3.2933
```

5. Učeći agenti (Learning Agents)

Učeći agenti predstavljaju najnapredniju klasu agenata u vještačkoj inteligenciji. Njihova osnovna karakteristika je sposobnost da kontinuirano unapređuju svoje ponašanje na osnovu iskustva i podataka prikupljenih iz okruženja. Za razliku od klasičnih agenata koji rade po unaprijed definisanim pravilima, učeći agenti adaptivno prilagođavaju svoje odluke promjenama u sistemu.

Struktura učećeg agenta

Tipična arhitektura učećeg agenta sastoji se od četiri osnovne komponente:

- **Learning element (element učenja)** – odgovoran za modifikaciju internog modela i poboljšanje strategije odlučivanja na osnovu novih podataka.
- **Performance element (element izvršavanja)** – donosi konkretne odluke i bira akcije na osnovu trenutne politike ponašanja.
- **Critic (kritičar)** – analizira rezultat izvršenih akcija i daje povratnu informaciju u obliku nagrade ili kazne.
- **Problem generator (generator problema)** – podstiče istraživanje novih strategija ponašanja radi pronalaženja boljih rješenja.

Algoritmi koje koriste učeći agenti

Učeći agenti se oslanjaju na napredne algoritme mašinskog učenja, među kojima se najčešće koriste:

- **Neuronske mreže** – omogućavaju modeliranje složenih nelinearnih odnosa između ulaznih i izlaznih veličina sistema.
- **Reinforcement Learning (učenje poticanjem)** – omogućava agentu da uči optimalne strategije kroz interakciju sa okruženjem na osnovu sistema nagrada i kazni.
- **Genetski algoritmi** – koriste principe prirodne selekcije za optimizaciju parametara modela i ponašanja agenta.

Zadaci za vježbanje

Zadatak1

Razviti učećeg agenta zasnovanog na neuronskoj mreži koji na osnovu izmjerenih ulaznih veličina sunčeve radijacije (G), temperature (T) i stanja napunjenošći baterije (SOC) donosi odluku o upravljačkom signalu duty cycle za DC-DC pretvarač u fotonaponskom sistemu. Cilj agenta je da održi stabilan izlazni napon sistema na vrijednosti od 12 V. Agent mora naučiti vezu između ulaznih parametara i potrebnog upravljačkog signala na osnovu dostupnih podataka.

```
clc; clear; close all;

%% 1. Trening podaci (simulirani)
% Ulazi: [G; T; SOC]
G = linspace(200,1000,100);
T = linspace(10,40,100);
SOC = linspace(0.3,1,100);

X = [G; T; SOC];

% Željeni izlaz - duty cycle (simulacija)
Y = 0.4 + 0.0005*G - 0.003*T + 0.2*SOC;

%% 2. Kreiranje učećeg agenta (neuronske mreže)
net = feedforwardnet(10);
net.trainParam.epochs = 300;

%% 3. Treniranje agenta
net = train(net, X, Y);

%% 4. Testiranje agenta
testInput = [800; 25; 0.7];
duty = net(testInput);

disp("Izlaz agenta - duty cycle:");
disp(duty);
```

Command Window

```
Izlaz agenta - duty cycle:
0.8812
```

Zadatak2

Kreirati i naučiti MLP neuronsku mrežu da predviđa cijenu stana na osnovu njegovih karakteristika. Napraviti ulaznu matricu X i ciljnu vrijednost Y (simulirani podaci), pa testirati mrežu na novim podacima i prikazati predikcije i stvarne vrijednosti.

Ulazi (features):

1. Površina stana (m^2)
2. Broj soba
3. Sprat
4. Starost zgrade (godine)

Izlaz (target): Cijena stana (u hiljadama KM)

```
clc; clear; close all;

%% Simulirani podaci
% Broj uzoraka
N = 100;

% Ulazi: Površina, Broj soba, Sprat, Starost zgrade
Povrsina = randi([40,150],1,N);      % m2
Sobe     = randi([1,5],1,N);         % broj soba
Sprat    = randi([1,10],1,N);        % sprat
Starost  = randi([0,50],1,N);        % godine

X = [Povrsina; Sobe; Sprat; Starost];

% Ciljna cijena (simulirana funkcija + šum)
Y = 50 + 1.2*Povrsina + 10*Sobe + 2*Sprat - 0.5*Starost + 5*randn(1,N);

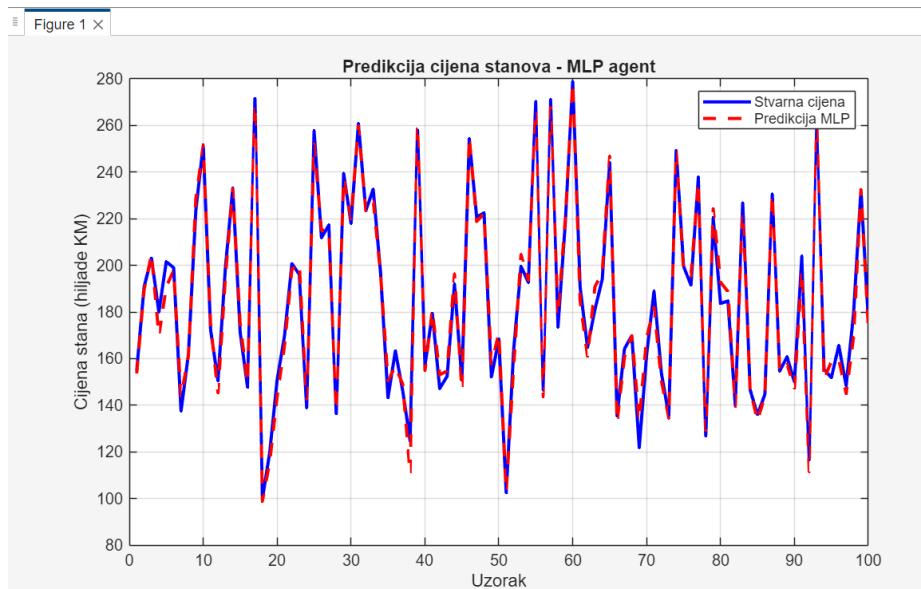
%% Kreiranje i treniranje MLP mreže
net = feedforwardnet(10);    % 1 skriveni sloj sa 10 neurona
net.trainParam.epochs = 500;
net = train(net, X, Y);

%% Testiranje na novim podacima
testX = [80; 3; 2; 10];    % primjer stana
predPrice = net(testX);

disp("Predviđena cijena stana (hiljade KM):");
disp(predPrice);

%% Prikaz rezultata: Predikcije vs stvarne vrijednosti
Y_pred = net(X);

figure;
plot(Y, 'b', 'LineWidth',2); hold on;
plot(Y_pred, 'r--', 'LineWidth',2);
grid on;
xlabel('Uzorak');
ylabel('Cijena stana (hiljade KM)');
title('Predikcija cijena stanova - MLP agent');
legend('Stvarna cijena', 'Predikcija MLP');
```



Zadatak3 Predikcija uključivanja ventilatora pomoću MLP mreže

Potrebno je automatski upravljati ventilatorom na osnovu temperature. Ventilator se uključuje kada temperatura prelazi prag od 30°C i isključuje kada temperatura padne ispod praga. Potrebno je kreirati MLP (multilayer perceptron) neuronsku mrežu koja na osnovu ulaznog signala temperature tokom dana predviđa stanje ventilatora ($0 = \text{OFF}$, $1 = \text{ON}$). Mreža treba naučiti vezu između temperature i uključivanja ventilatora i pravilno predviđati kada ventilator treba da se uključi ili isključi.

Potrebno je:

1. Generisati deterministički niz temperatura koji simulira promjene tokom dana: od minimalne temperature od 15°C do maksimalne od 35°C .
2. Definisati stvarno stanje ventilatora:
 - o $1 (\text{ON})$ ako je temperatura $\geq 30^{\circ}\text{C}$
 - o $0 (\text{OFF})$ ako je temperatura $< 30^{\circ}\text{C}$
3. Kreirati MLP sa dva skrivena sloja (10 i 5 neurona) i ReLU aktivacijom u skrivenim slojevima. Izlazni sloj koristi logsig aktivaciju.
4. Trenirati mrežu na svim uzorcima temperature.
5. Predikcija mreže treba da bude binarna (0 ili 1) i da što preciznije prati stvarno stanje ventilatora.
6. Prikažite rezultate kroz graf:
 - o Subplot 1: temperatura tokom dana sa crvenom isprekidanom linijom koja označava prag od 30°C .
 - o Subplot 2: stvarno stanje ventilatora i predikcija MLP-a.

Predikcija mreže treba da bude logična i precizna (tačnost $> 95\%$). Može se koristiti prag 0.5 za konverziju izlaza MLP mreže u binarni signal.

```

clc; clear; close all;

%% 1. Generisanje podataka
N = 500; % veći broj uzoraka
t = linspace(0,24,N); % vrijeme u satima

% Temperatura tokom dana (deterministički sinus)
T = 25 + 10*sin(pi*(t-6)/12); % od 15 do 35°C

% Stvarno stanje ventilatora (1=ON, 0=OFF)
Fan = double(T >= 30);

% Ulazna matrica (samo temperatura za jednostavnost)
X = T;

%% 2. Kreiranje MLP
hiddenLayerSize = [10 5]; % dva sloja: 10 neurona + 5 neurona
net = feedforwardnet(hiddenLayerSize);

% Aktivacija slojeva: ReLU (poslin) za bolje ucenje naglih promjena
for i = 1:length(net.layers)-1
    net.layers{i}.transferFcn = 'poslin';
end
% Izlazni sloj: logsig za 0-1
net.layers{end}.transferFcn = 'logsig';

% Trening
net.trainFcn = 'trainlm';
net.trainParam.epochs = 1000;

%% 3. Treniranje mreže
net = train(net, X, Fan);

%% 4. Predikcija
Y_pred = net(X);
Y_class = double(Y_pred > 0.5); % prag 0.5 daje binarni izlaz

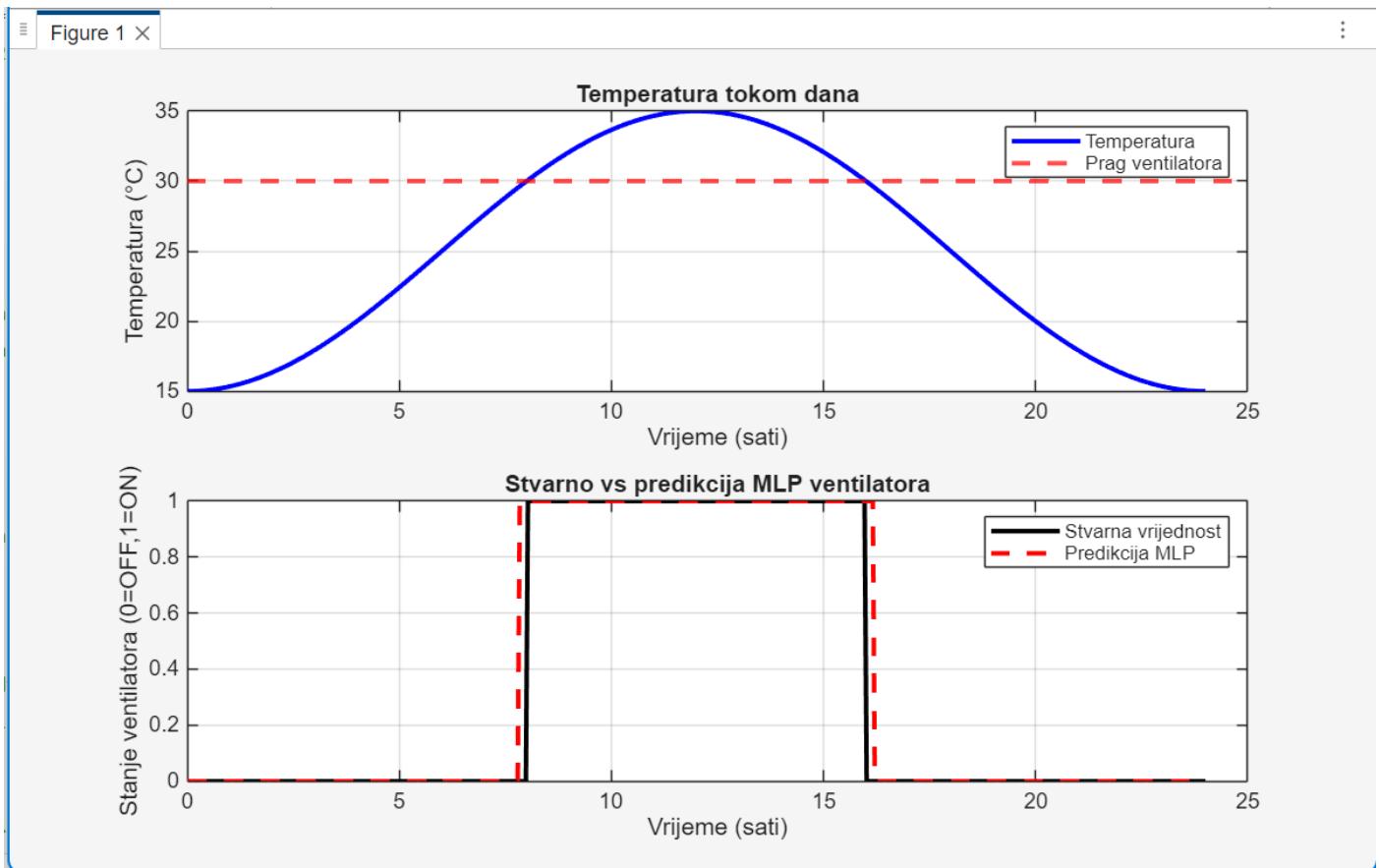
%% 5. Tačnost
accuracy = sum(Y_class == Fan)/N * 100;
fprintf('Tačnost MLP: %.2f %%\n', accuracy);

%% 6. Grafički prikaz
figure;

% Subplot 1: temperatura
subplot(2,1,1);
plot(t, T,'b-','LineWidth',2); hold on;
yline(30,'r--','LineWidth',2); % prag ventilatora
grid on;
xlabel('Vrijeme (sati)');
ylabel('Temperatura (°C)');
title('Temperatura tokom dana');
legend('Temperatura','Prag ventilatora');

% Subplot 2: ventilator
subplot(2,1,2);
plot(t, Fan,'k-','LineWidth',2); hold on;
plot(t, Y_class,'r--','LineWidth',2);
grid on;
xlabel('Vrijeme (sati)');
ylabel('Stanje ventilatora (0=OFF,1=ON)');
title('Stvarno vs predikcija MLP ventilatora');
legend('Stvarna vrijednost','Predikcija MLP');

```



Zadatak 4

Razviti MLP neuronsku mrežu koja predviđa sentiment:

- 1 = pozitivna recenzija
- 0 = negativna recenzija

Tekst datoteka **reviews.txt** (20 recenzija)

*I absolutely love this product, it works perfectly and exceeded my expectations
The item was terrible, I am very disappointed and would not recommend it
Fantastic quality, I am very happy with my purchase and everything works great
This is the worst experience I have ever had, very poor design and functionality
I am impressed, excellent performance and perfect for my needs
Awful, it broke after two uses and I hate it
Good value for money, really satisfied with the product
Terrible service, the item arrived damaged and support was awful
I highly recommend this, fantastic and awesome product
I dislike this product, very bad quality and poor performance
Perfect, just what I needed and works as described
I am very disappointed, it did not meet my expectations
Great item, excellent design and I love using it
Worst purchase ever, terrible quality and I hate it
Amazing product, very happy with the results and recommend it
Poor build, bad materials and disappointing overall
I am satisfied, good quality and works perfectly*

*Awful experience, would never buy again
Fantastic, excellent, and perfect, very happy
Terrible, worst item I have bought, do not recommend*

```
clc; clear; close all;

%% 1. Ucitavanje fajla
fid = fopen('reviews.txt','r');
C = textscan(fid, '%s', 'Delimiter', '\n');
fclose(fid);
reviews = C{1};
N = length(reviews);

%% 2. Lista pozitivnih i negativnih rijeci
positiveWords =
{'good', 'excellent', 'fantastic', 'love', 'great', 'happy', 'recommend',
'awesome', 'perfect'};
negativeWords =
{'bad', 'poor', 'terrible', 'hate', 'awful', 'worst', 'disappointed'};

%% 3. Kreiranje ulaznih podataka
X = zeros(2,N); % 2 ulaza: broj pozitivnih i negativnih
rijeci
Y = zeros(1,N); % ciljni izlaz: 0=negativno, 1=pozitivno

for i = 1:N
    text = lower(reviews{i});
    words = split(text);
    posCount = sum(ismember(words, positiveWords));
    negCount = sum(ismember(words, negativeWords));
    X(:,i) = [posCount; negCount];

    % Pravilo za cilj
    if posCount > negCount
        Y(i) = 1;
    else
        Y(i) = 0;
    end
end

%% 4. Kreiranje i trening MLP
net = feedforwardnet(5); % skriveni sloj sa 5 neurona
net.trainParam.epochs = 500;
net = train(net, X, Y);

%% 5. Predikcija
Y_pred = net(X);
Y_class = double(Y_pred > 0.5);
```

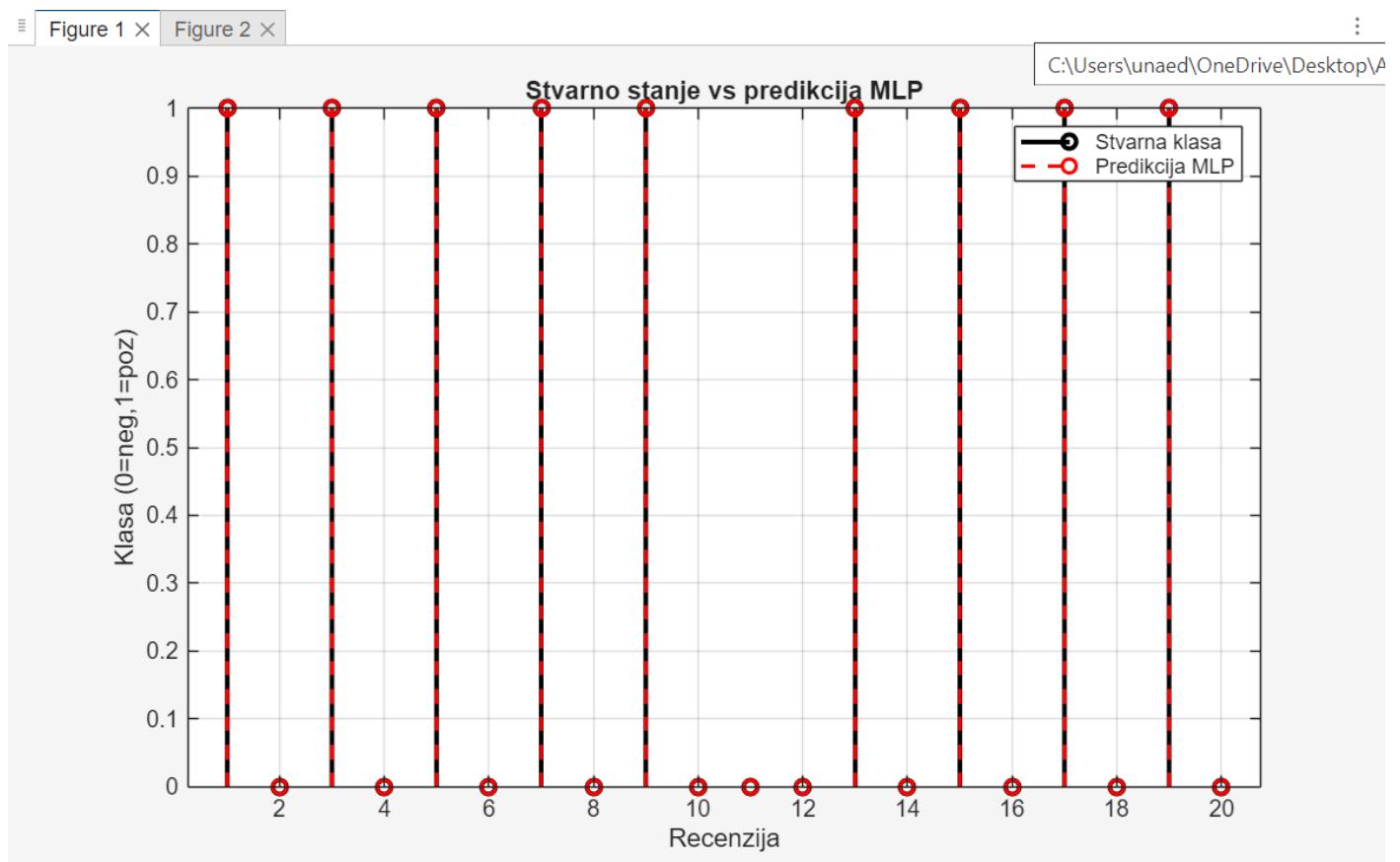
```

%% 6. Tacnost
accuracy = sum(Y_class == Y) / N * 100;
fprintf('Tacnost MLP: %.2f %%\n', accuracy);

%% 7. Graf predikcija vs stvarno stanje
figure;
stem(1:N, Y, 'k', 'LineWidth', 2); hold on;
stem(1:N, Y_class, 'r--', 'LineWidth', 1.5);
xlabel('Recenzija'); ylabel('Klasa (0=neg,1=poz)');
title('Stvarno stanje vs predikcija MLP');
legend('Stvarna klasa', 'Predikcija MLP');
grid on;

```

Od ukupnog broja 20 recenzija:



UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

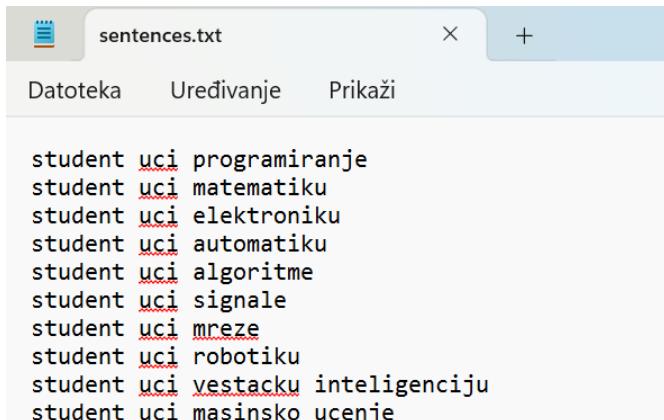
VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

Primjer 1

Razviti jednostavnu neuronsku mrežu koja na osnovu niza tekstualnih rečenica uči redoslijed i obrazac pojavljivanja riječi, te na osnovu naučenog modela generiše novu rečenicu koja logički slijedi dati niz. Ulaz u sistem je niz kratkih rečenica zapisanih u .txt fajlu. Svaka rečenica ima sličnu strukturu i pripada istom tematskom kontekstu. Neuronska mreža treba da nauči koja riječ najčešće slijedi nakon prethodne, te da na osnovu toga predviđa naredni tekstualni element.

Ulagani podaci



The screenshot shows a text editor window with the file 'sentences.txt' open. The content of the file is as follows:

```
student uci programiranje
student uci matematiku
student uci elektroniku
student uci automatiku
student uci algoritme
student uci signale
student uci mreze
student uci robotiku
student uci vestacku inteligenciju
student uci masinsko ucenje
```

```
clc; clear; close all;

%% 1. Učitavanje teksta
fid = fopen('sentences.txt','r');
C = textscan(fid, '%s', 'Delimiter', '\n');
fclose(fid);
sentences = C{1};

%% 2. Ekstrakcija riječi
allWords = {};
for i = 1:length(sentences)
    words = split(lower(sentences{i}));
    allWords = [allWords; words];
end

uniqueWords = unique(allWords);
numWords = length(uniqueWords);

%% 3. Mapiranje riječi u indekse
word2idx = containers.Map(uniqueWords,1:numWords);
idx2word = uniqueWords;

%% 4. Kreiranje ulaza i izlaza
% Ulaz: [student uci] → Izlaz: oblast
X = [];
Y = [];

for i = 1:length(sentences)
    words = split(lower(sentences{i}));
    inputVec = zeros(numWords,1);
    inputVec(word2idx(words{1})) = 1;
    inputVec(word2idx(words{2})) = 1;

    outputVec = zeros(numWords,1);
    outputVec(word2idx(words{3})) = 1;

    X = [X inputVec];
    Y = [Y outputVec];
end
```

```

%% 5. Neuronska mreža
net = feedforwardnet(15);
net.trainParam.epochs = 500;
net = train(net,X,Y);

%% 6. Predikcija nove rečenice
testInput = zeros(numWords,1);
testInput(word2idx('student')) = 1;
testInput(word2idx('uci')) = 1;

prediction = net(testInput);
[~,idx] = max(prediction);

fprintf('Predložena nova rečenica:\n');
fprintf('student uci %s\n', idx2word{idx});

```

Izlaz:

	Command Window
	Predložena nova rečenica: student uci automatiku >>

Primjer 2

Razviti MLP neuronsku mrežu koja predviđa sentiment:

- 1 = pozitivna recenzija
- 0 = negativna recenzija

Tekst datoteka **reviews.txt** (20 recenzija)

*I absolutely love this product, it works perfectly and exceeded my expectations
The item was terrible, I am very disappointed and would not recommend it
Fantastic quality, I am very happy with my purchase and everything works great
This is the worst experience I have ever had, very poor design and functionality
I am impressed, excellent performance and perfect for my needs
Awful, it broke after two uses and I hate it
Good value for money, really satisfied with the product
Terrible service, the item arrived damaged and support was awful
I highly recommend this, fantastic and awesome product
I dislike this product, very bad quality and poor performance
Perfect, just what I needed and works as described
I am very disappointed, it did not meet my expectations
Great item, excellent design and I love using it
Worst purchase ever, terrible quality and I hate it
Amazing product, very happy with the results and recommend it
Poor build, bad materials and disappointing overall
I am satisfied, good quality and works perfectly
Awful experience, would never buy again
Fantastic, excellent, and perfect, very happy
Terrible, worst item I have bought, do not recommend*

```

clc; clear; close all;

%% 1. Ucitavanje fajla
fid = fopen('reviews.txt','r');
C = textscan(fid, '%s', 'Delimiter', '\n');
fclose(fid);
reviews = C{1};
N = length(reviews);

%% 2. Lista pozitivnih i negativnih rijeci
positiveWords =
{'good', 'excellent', 'fantastic', 'love', 'great', 'happy', 'recommend',
'awesome', 'perfect'};
negativeWords =
{'bad', 'poor', 'terrible', 'hate', 'awful', 'worst', 'disappointed'};

%% 3. Kreiranje ulaznih podataka
X = zeros(2,N); % 2 ulaza: broj pozitivnih i negativnih rijeci
Y = zeros(1,N); % ciljni izlaz: 0=negativno, 1=pozitivno

for i = 1:N
    text = lower(reviews{i});
    words = split(text);
    posCount = sum(ismember(words,positiveWords));
    negCount = sum(ismember(words,negativeWords));
    X(:,i) = [posCount; negCount];

    % Pravilo za cilj
    if posCount > negCount
        Y(i) = 1;
    else
        Y(i) = 0;
    end
end

%% 4. Kreiranje i trening MLP
net = feedforwardnet(5); % skriveni sloj sa 5 neurona
net.trainParam.epochs = 500;
net = train(net,X,Y);

%% 5. Predikcija
Y_pred = net(X);
Y_class = double(Y_pred > 0.5);

%% 6. Tacnost
accuracy = sum(Y_class == Y)/N * 100;
fprintf('Tacnost MLP: %.2f %%\n', accuracy);

%% 7. Graf predikcija vs stvarno stanje

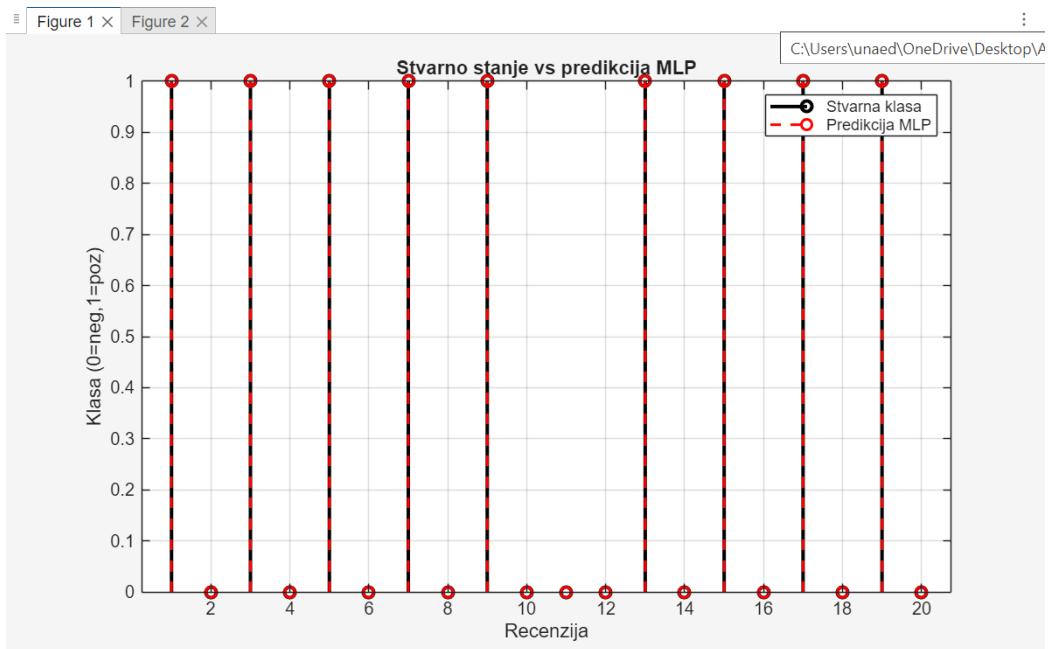
```

```

figure;
stem(1:N,Y,'k','LineWidth',2); hold on;
stem(1:N,Y_class,'r--','LineWidth',1.5);
xlabel('Recenzija'); ylabel('Klasa (0=neg,1=poz)');
title('Stvarno stanje vs predikcija MLP');
legend('Stvarna klasa','Predikcija MLP');
grid on;

```

Od ukupnog broja 20 recenzija:



Primjer 3

Razviti konvolucionu neuronsku mrežu (CNN) koja na osnovu ulazne slike prepoznaje da li se na slici nalazi horizontalna linija ili vertikalna linija. Ulaz u sistem su jednostavne binarne slike dimenzije 20×20 piksela. Izlaz neuronske mreže je klasa slike: 0 – vertikalna linija (crvena) i 1 – horizontalna linija (plava)

```

clc; clear; close all;

% 1. Parametri
numImages = 200;
imgSize = 20;

% 2. Generisanje RGB slika sa tankim linijama
X = zeros(imgSize,imgSize,3,numImages);
Y = categorical(zeros(numImages,1));

for i = 1:numImages
    img = zeros(imgSize,imgSize,3);

    if mod(i,2) == 0
        % Horizontalna linija - PLAVA
        img(10,:,:)=1; % B kanal
        Y(i) = categorical(1); % Klasa 1
    else
        % Vertikalna linija - CRVENA
        img(:,10,1)=1; % R kanal
        Y(i) = categorical(0); % Klasa 0
    end
end

```

```

% Blagi šum (realističniji ulaz)
img = img + 0.05*randn(size(img));
img = max(min(img,1),0);

X(:,:,:,i) = img;
end

%% 3. Podjela na trening i test skup
idx = randperm(numImages);
trainIdx = idx(1:150);
testIdx = idx(151:end);

XTrain = X(:,:,:,:,trainIdx);
YTrain = Y(trainIdx);

XTest = X(:,:,:,:,testIdx);
YTest = Y(testIdx);

%% 4. CNN arhitektura
layers = [
    imageInputLayer([imgSize imgSize 3])

    convolution2dLayer(3,8,'Padding','same')
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    reluLayer

    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
];
];

%% 5. Opcije treninga
options = trainingOptions('adam', ...
    'MaxEpochs',10, ...
    'MiniBatchSize',16, ...
    'Verbose',false);

%% 6. Treniranje mreže
net = trainNetwork(XTrain,YTrain,layers,options);

%% 7. Testiranje
YPred = classify(net,XTest);
accuracy = sum(YPred == YTest)/numel(YTest)*100;

fprintf('Tačnost CNN-a: %.2f %%\n', accuracy);

%% 8. Vizualizacija test slika i predikcije
figure;
for i = 1:6
    subplot(2,3,i);
    imshow(XTest(:,:,:,i));
    title(['Predikcija: ' char(YPred(i))]);
end

```

Figure 1 X

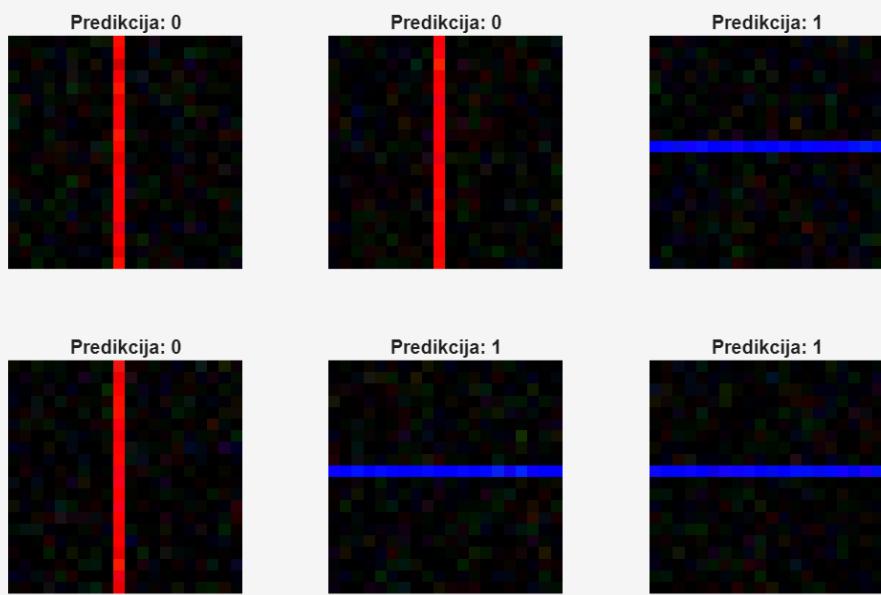
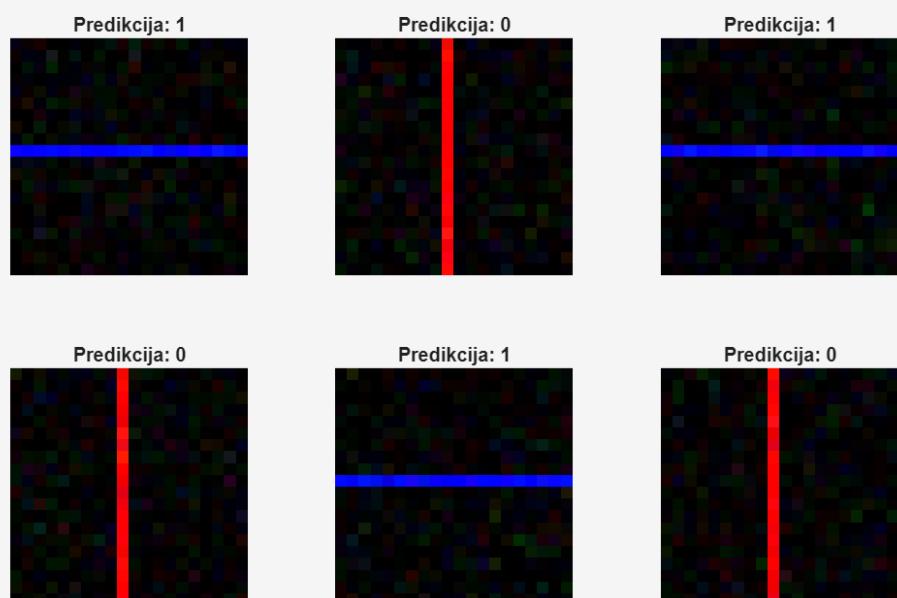


Figure 1 X



Primjer 4

Dat je skup slika geometrijskih oblika organizovan u direktorije po klasama (krug, kvadrat, trougao, itd.). Razviti konvolucionu neuronsku mrežu (CNN) u MATLAB-u koja na osnovu ulazne slike prepoznaće kojem geometrijskom obliku slika pripada.

```
% CNN za prepoznavanje geometrijskih oblika i prikaz predikcija
clc; clear; close all;

% Folder dataset-a
datasetFolder = fullfile(pwd, 'Shapes_Dataset');

% Učitavanje slika
imds = imageDatastore(datasetFolder, ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

% Promjena veličine slika na 64x64
inputSize = [64 64 3];
imds.ReadFcn = @(x) imresize(imread(x), inputSize(1:2));

% Podjela na trening i test skup
[imdsTrain, imdsTest] = splitEachLabel(imds, 0.8, 'randomized');

%% CNN arhitektura
layers = [
    imageInputLayer(inputSize)

    convolution2dLayer(3,8, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3,16, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3,32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(numel(unique(imds.Labels)))
    softmaxLayer
    classificationLayer];

%% Opcije treniranja
options = trainingOptions('adam', ...
    'MaxEpochs',5, ... % povećati po potrebi
    'MiniBatchSize',64, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsTest, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');

%% Treniranje mreže
net = trainNetwork(imdsTrain, layers, options);

%% Testiranje i tačnost
YPred = classify(net, imdsTest);
YTest = imdsTest.Labels;
accuracy = sum(YPred == YTest)/numel(YTest);
disp(['Tačnost na test skupu: ', num2str(accuracy*100), '%']);

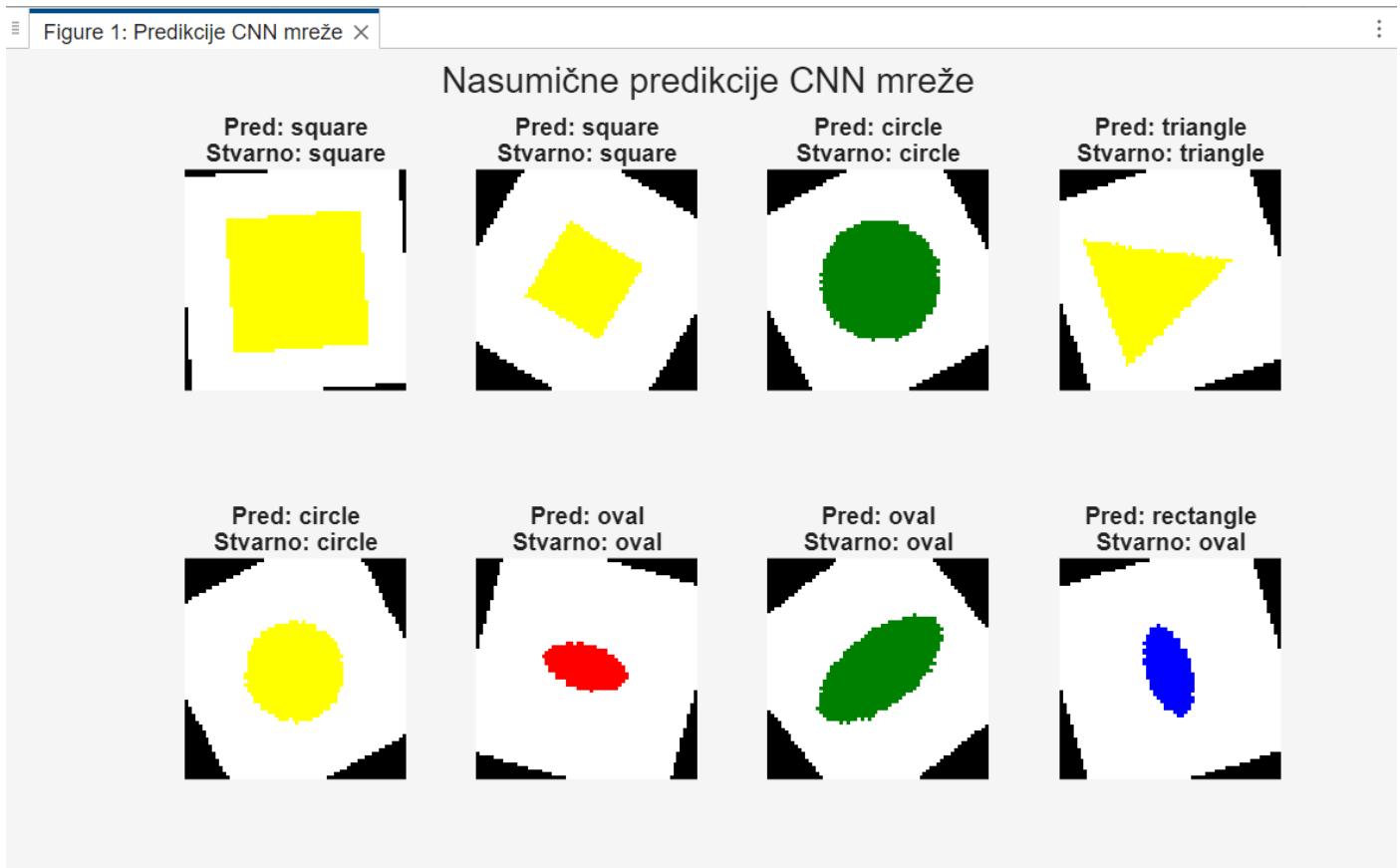
%% Prikaz nasumičnih 8 predikcija
numExamples = 8;
```

```

idx = randperm(numel(imdsTest.Files), numExamples);

figure('Name','Predikcije CNN mreže');
for i = 1:numExamples
    img = readimage(imdsTest, idx(i));
    subplot(2,4,i);
    imshow(img);
    title({['Pred: ', char(YPred(idx(i)))], ['Stvarno: ', char(YTest(idx(i)))]}, ...
        'FontSize',10);
end
sgtitle('Nasumične predikcije CNN mreže');

```



Command Window

Tačnost na test skupu: 80.2%