Univerzitetna založba
Univerze v Mariboru

STROJNO UČENJE ZA INŽENIRJE

koncepti, primeri in uporaba v okolju MATLAB

Janez Gotlih Miran Brezočnik

Univerza v Mariboru
Fakulteta za strojništvo

Strojno učenje za inženirje:

Koncepti, primeri in uporaba v okolju MATLAB

Avtorja
Janez Gotlih
Miran Brezočnik

November 2025

Naslov *Title*
Strojno učenje za inženirje
Machine Learning for Engineers

Podnaslov *Subtitle*
Koncepti, primeri in uporaba v okolju MATLAB
Concepts, Examples, and Applications in MATLAB

Avtorja *Authors*
Janez Gotlih
(Univerza v Mariboru, Fakulteta za strojništvo)

Miran Brezočnik
(Univerza v Mariboru, Fakulteta za strojništvo)

Recenzija *Review*
Mirko Ficko
(Univerza v Mariboru, Fakulteta za strojništvo)

Simon Klančnik
(Univerza v Mariboru, Fakulteta za strojništvo)

Jezikovni pregled *Language editing*
Danica Gotlih

Tehnična urednika *Technical editors*
Marina Bajić
(Univerza v Mariboru, Univerzitetna založba)

Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Oblikovanje ovitka *Cover designer*
Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Grafika na ovitku *Cover graphic*
Striped textile, foto: Scott Wemb, unsplash.com, 2020

Založnik *Published by*
Univerza v Mariboru
Univerzitetna založba
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

Izdajatelj *Issued by*
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova ulica 17, 2000 Maribor, Slovenija
https://www.fs.um.si/, fs@um.si

Izdaja *Edition*
Prva izdaja

Vrsta publikacije *Publication type*
E-knjiga

Dostopno na *Available at*
https://press.um.si/index.php/ump/catalog/book/1075

Izdano *Published at*
Maribor, Slovenija, november 2025

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004.42:51

GOTLIH, Janez
Strojno učenje za inženirje [Elektronski vir] : koncepti, primeri in uporaba v okolju MATLAB / Janez Gotlih, Miran Brezočnik. - 1. izd. - E-knjiga. - Maribor : Univerza v Mariboru, Univerzitetna založba, 2025

Način dostopa (URL):
https://press.um.si/index.php/ump/catalog/book/1075

ISBN 978-961-299-078-7 (PDF)
doi: 10.18690/um.fs.10.2025
COBISS.SI-ID 256145411

ISBN 978-961-299-078-7 (pdf)

DOI https://doi.org/10.18690/um.fs.10.2025

Cena *Price* Brezplačni izvod

Odgovorna oseba založnika *For publisher* Prof. dr. Zdravko Kačič, rektor Univerze v Mariboru

Citiranje *Attribution* Gotlih, J., Brezočnik, M.(2025). *Strojno učenje za inženirje: koncepti, primeri in uporaba v okolju MATLAB.* Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.fs.10.2025

Univerzitetna založba
Univerze v Mariboru

Kazalo

Univerzitetna založba
Univerze v Mariboru

1 Uvod v strojno učenje

Strojno učenje (angl. Machine Learning) je področje umetne inteligence, ki se ukvarja z razvojem algoritmov, ki omogočajo računalnikom učenje iz podatkov in sprejemanje odločitev brez eksplicitnega programiranja pravil. Strojno učenje se vedno bolj uveljavlja kot ključna tehnologija v sodobnem inženirstvu, saj omogoča razvoj inteligentnih sistemov za analizo, optimizacijo in avtomatizacijo procesov. Med temeljnimi pristopi strojnega učenja so nadzorovano učenje, nenadzorovano učenje, učenje z okrepitvijo in prenosno učenje.

Nadzorovano učenje temelji na označenih podatkih in vključuje metode, ki omogočajo napovedovanje vrednosti (regresija) ali razvrščanje v razrede (klasifikacija). V inženirskih aplikacijah se nadzorovano učenje uporablja na primer za napovedovanje stanja sistemov, ocenjevanje učinkovitosti procesov ali za avtomatizirano zaznavanje napak. Med pogostimi metodami so odločitvena drevesa, metode podpornih vektorjev, nevronske mreže in modeli z Gaussovimi procesi.

Nenadzorovano učenje išče vzorce v neoznačenih podatkih. Vključuje gručenje (npr. prepoznavanje vzorcev v podatkih) in zmanjševanje dimenzionalnosti (npr. poenostavitev kompleksnih podatkov za analizo in vizualizacijo). Med pogostimi pristopi so K-sredinsko gručenje, metoda glavnih komponent PCA in metoda za nelinearno zmanjševanje dimenzionalnosti t-SNE.

Učenje z okrepitvijo temelji na interakciji z okoljem, kjer agent pridobiva izkušnje s preizkušanjem dejanj in učenjem iz prejetih nagrad ali kazni. Uporablja se predvsem v primerih, kjer je cilj optimizirati zaporedje odločitev, kot so vodenje dinamičnih sistemov, prilagodljivo odločanje, načrtovanje nalog ali optimizacija virov.

Poleg glavnih kategorij strojnega učenja pa se je uveljavilo tudi prenosno učenje, ki omogoča uporabo že naučenih modelov za reševanje sorodnih nalog z manj podatki in krajšim časom učenja. Prenosno učenje se v inženirstvu uporablja pri strojnem vidu, obdelavi signalov ter prenosu znanja med podobnimi procesi ali napravami.

Vsi primeri v teh skriptih so pripravljeni in testirani v okolju MATLAB R2024a. Za izvajanje nalog in uporabo grafičnih orodij so potrebni: Statistics and Machine Learning Toolbox, Deep Learning Toolbox, Reinforcement Learning Toolbox, Predictive Maintenance Toolbox, Image Processing Toolbox.

Univerzitetna založba
Univerze v Mariboru

2 Nadzorovano učenje

Nadzorovano učenje (angl. Supervised Learning) se uporablja za napovedovanje rezultatov na podlagi znanih vhodnih podatkov in pripadajočih izhodov. Na primer, napovedovanje življenjske dobe strojev na podlagi zgodovinskih meritev.

Glavni podskupini sta razvrščanje in regresija. Pri tem regresija napoveduje zvezne (numerične) vrednosti, medtem ko klasifikacija razvršča podatke v diskretne razrede ali kategorije.

Razvrščanje ali klasifikacija (angl. Classification) se uporablja, kadar želimo razvrstiti vzorce v končne kategorije. V inženirstvu se uporablja pri vizualni kontroli kakovosti (npr. razvrščanje izdelkov na dobre/slabe), diagnostiki napak v proizvodnih sistemih, prepoznavanju anomalij v delovanju strojev, razvrščanju materialov ali komponent glede na senzorske podatke ipd.

Najpogosteje uporabljene metode razvrščanja so:

- Naivni Bayesov klasifikator (angl. Naive Bayes classifier): metoda temelji na verjetnosti in predpostavki neodvisnosti značilk.
- Odločitvena drevesa (angl. Decision Trees): gradijo strukturo odločanja na osnovi pogojev.

- Metoda podpornih vektorjev (angl. Support Vector Machine oz. SVM): poišče optimalno mejo med razredi.
- Naključni gozd (angl. Random Forest): kombinacija več dreves za večjo robustnost.
- K-najbližjih sosedov (angl. k-Nearest Neighbors oz. KNN): klasifikacija na podlagi podobnosti z bližnjimi primeri.
- Umetne nevronske mreže (angl. Artificial Neural Network oz. ANN): učinkovite pri kompleksnih vhodih, npr. slikah.
- Logistična regresija (angl. Logistic Regression): napovedovanje binarnega ali večrazrednega izida.

Regresija (angl. Regression) se uporablja za napovedovanje zveznih (numeričnih) vrednosti. V inženirstvu jo uporabljamo za oceno obrabe orodij, življenjske dobe komponent, energetskih potreb, mehanskih lastnosti materialov ter za kalibracijo senzorjev in napovednih modelov.

Najpogosteje uporabljene regresijske metode so:

- Linearna regresija (angl. Linear Regression): najpreprostejši model za linearno odvisnost.
- Regresija z nevronskimi mrežami (angl. Neural Network Regression): zajema kompleksne nelinearne odnose.
- Regresija s podpornimi vektorji (angl. Support Vector Regression oz. SVR): metoda, robustna na izjeme.
- Odločitvena regresijska drevesa (angl. Decision Regression Trees): podobna klasifikacijski metodi, a namesto razreda napoveduje zvezno vrednost.
- Lasso regresija (angl. Lasso Regression): znižuje kompleksnost modela z ničelnimi koeficienti.
- Ridge regresija (angl. Ridge Regression): metoda omejuje rast koeficientov za preprečevanje prenaučenosti (angl. overfitting).
- Regresija z Gaussovimi procesi (angl. Gaussian Process Regression oz. GPR): verjetnostni pristop, ki se uporablja v eksperimentalnem modeliranju in optimizaciji.

Za lažje razumevanje razlik med klasifikacijo in regresijo je v preglednici (Tabela 1) povzeta primerjava obeh pristopov glede na vhodne in izhodne podatke, tipične inženirske primere ter pogosto uporabljene metode.

Tabela 1: Primerjava klasifikacije in regresije v nadzorovanem učenju

Področje	Klasifikacija	Regresija
Vhodni podatki	Značilke (npr. senzorski podatki, slike, meritve)	Značilke (npr. senzorski podatki, meritve)
Izhodna spremenljivka	Diskretna vrednost (razred, kategorija)	Zvezna (numerična) vrednost
Primer v inženirstvu	Razvrščanje izdelkov na dobre/slabe; diagnostika napak; prepoznavanje materialov	Napoved obrabe orodja; življenjske dobe komponente; energetskih potreb. mehanskih lastnosti materialov
Tipične metode	Naivni Bayes; Odločitvena drevesa; SVM; Naključni gozd; KNN; Nevronske mreže; Logistična regresija	Linearna regresija; Nevronske mreže; SVR; Regresijska drevesa; Lasso; Ridge; Regresija z Gaussovimi procesi (GPR)

2.1 Primer: Nadzorovano učenje za regresijo

V tem primeru bomo oblikovali model regresije z Gaussovimi procesi (GPR) za napoved površine vara na podlagi dveh vhodnih parametrov: toka in podajalne hitrosti. GPR je verjetnostna metoda regresije, ki na podlagi vhodnih podatkov omogoča napoved vrednosti in oceno negotovosti napovedi. Metoda GPR je primerna za manjše nize eksperimentalnih podatkov.

2.1.1 Priprava podatkov

Za to nalogo bomo uporabili podatke iz poglavja 8.1 (Priloga 1). Na podlagi podatkov ustvarite datoteko CSV z imenom *povrsina_vara.csv*.

- Povlecite in spustite datoteko *povrsina_vara.csv* v polje **Files** in počakajte, da se podatki prenesejo.
- Kliknite **OK**.

Priprava dokumenta in branje podatkov

- Ustvarite nov skript z imenom *povrsina_vara_GPR.m*.

- V odprt skript prilepite spodnjo kodo.

```
% 1. Preberi numerične podatke brez naslovov z vejico kot decimalnim
ločilom
tbl = readmatrix('povrsina_vara.csv', 'Range', 'A2:C51',
'DecimalSeparator', ',');
```

- Dodajte spodnje vrstice.

```
% 2. Razdeli stolpce
Tok     = tbl(:,1);
Hitrost = tbl(:,2);
Povrsina = tbl(:,3);
```

S tem smo ustvarili tri vektorje s podatki o toku (A), podajalni hitrosti (cm/min) in izmerjeni površini vara (mm^2).

2.1.2 Učenje modela GPR

- Model GPR naučite na celotnem naboru podatkov.

```
% 3. model GPR
X = [Tok, Hitrost];
y = Povrsina;

gprMdl = fitrgp(X, y);
```

Z uporabo ukaza *fitrgp* se model GPR (*gprMdl*) nauči napovedovati površino vara na podlagi toka in podajalne hitrosti (*X*) z uporabo dveh vhodnih spremenljivk in ustreznih rezultatov meritev površine (*y*).

2.1.3 Napoved in vrednotenje modela

- Ocenite natančnost modela z metrikama RMSE (koren srednje kvadratne napake) in R^2 (koeficient določnosti).

Namig:

RMSE izračunamo po formuli:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

R^2 pa izračunamo po formuli:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2},$$

kjer je:

- y_i dejanska vrednost,
- $\hat{y}_i$ napovedana vrednost,
- $\bar{y}_i$ povprečje dejanskih vrednosti,
- n število vzorcev.

```
% 4. Napoved in metriki
yPred = predict(gprMdl, X);

rmse = sqrt(mean((y - yPred).^2));
r2   = 1 - sum((y - yPred).^2) / sum((y - mean(y)).^2);

fprintf("RMSE: %.4f\n", rmse);
fprintf("R²: %.4f\n", r2);
```

Model z uporabo funkcije *predict* ustvari napovedi (*yPred*) za podane vhodne podatke *X*. Izračunata se metriki *rmse* (koren srednje kvadratne napake), ki meri povprečno odstopanje napovedi od dejanskih vrednosti, in *r2* (koeficient določnosti), ki kaže, kolikšen delež variabilnosti ciljne spremenljivke pojasni model. Na koncu funkcija *fprintf* prikaže izračunani *rmse* in *r2* v ukaznem oknu.

➢ Kodo zaženite s klikom na gumb **Run** ali s pritiskom tipke **F5**.

Rezultati modela dobro napoveduje površino vara glede na tok in podajalno hitrost, saj RMSE = 1,35 mm² pomeni nizko povprečno napako glede na razpon vrednosti (2–24 mm²). R^2 = 0,9156 kaže, da model pojasni več kot 91 % variance.

2.1.4 Vizualizacija napovedi

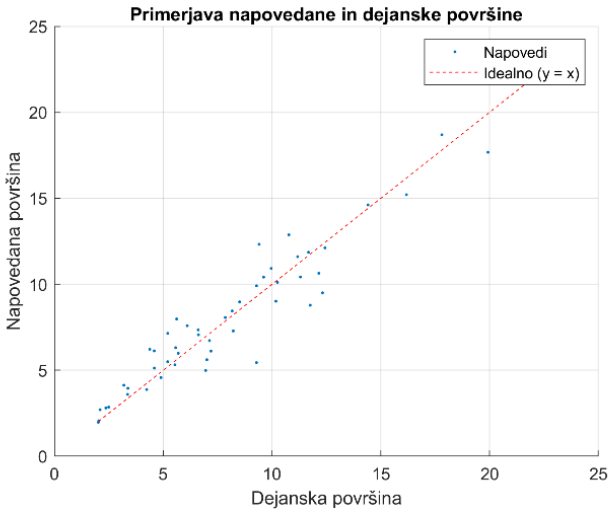
➢ Izrišite primerjalni graf med dejanskimi in napovedanimi vrednostmi.

```
% 5. Vizualizacija napovedi
figure;
scatter(y, yPred, '.');
xlabel('Dejanska površina'); ylabel('Napovedana površina');
title('Primerjava napovedane in dejanske površine');
hold on; plot([min(y), max(y)], [min(y), max(y)], 'r--');
legend('Napovedi', 'Idealno (y = x)');
grid on;
```

Za oceno kakovosti napovedi modela se ustvari raztreseni graf, kjer so na vodoravni osi prikazane dejanske vrednosti ciljne spremenljivke *y*, na navpični osi pa napovedane vrednosti *yPred*. Vsaka točka na grafu predstavlja en primer. Idealna napoved, pri kateri bi napoved popolnoma ustrezala dejanski vrednosti (y = x), je prikazana z rdečo črtkano diagonalo. Prisotnost točk ob tej premici pomeni dobro ujemanje napovedi z dejanskimi podatki. Legenda in mreža sta dodani za boljšo berljivost grafa ter hitrejšo vizualno presojo kakovosti modela.

➢ Kodo zaženite s klikom na gumb **Run** ali s pritiskom tipke **F5**.

Slika 1 prikazuje raztreseni graf napovedanih vrednosti površine vara glede na dejanske izmerjene vrednosti. Vsaka modra točka predstavlja en primer napovedi modela. Rdeča črtkana diagonala označuje idealno napoved ($y = x$), kjer bi se napoved popolnoma ujemala z dejansko vrednostjo. Večina točk leži blizu diagonale, kar kaže na dobro natančnost modela. Nekatera odstopanja so vidna pri večjih površinah, vendar je splošno prileganje ocenjeno kot zelo dobro (R^2 = 0,9156).



Slika 1: Primerjava napovedanih in dejanskih vrednosti površine vara (model GPR)

2.1.5 Površina napovedi

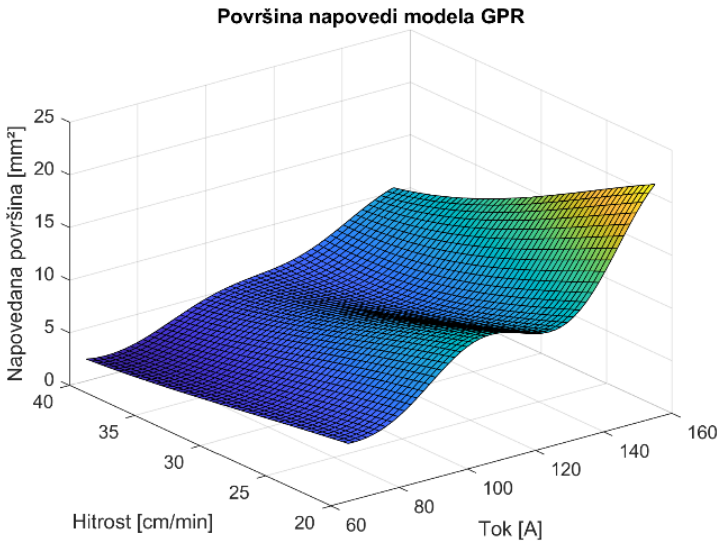
➢ Prikažite odzivno površino površine vara v odvisnosti od toka in hitrosti.

```
% 6. Odzivna površina
[tokGrid, hitrostGrid] = meshgrid( ...
    linspace(min(Tok), max(Tok), 50), ...
    linspace(min(Hitrost), max(Hitrost), 50));
Xsurf = [tokGrid(:), hitrostGrid(:)];
Ysurf = predict(gprMdl, Xsurf);
Zsurf = reshape(Ysurf, size(tokGrid));

figure;
surf(tokGrid, hitrostGrid, Zsurf);
xlabel('Tok [A]');
ylabel('Hitrost [cm/min]');
zlabel('Napovedana površina [mm²]');
title('GPR površina napovedi');
grid on;
```

Ta del kode pripravi in izriše 3D-površino napovedi modela. Model najprej ustvari mrežo vrednosti *tokGrid* in *hitrostGrid* s pomočjo funkcije *meshgrid*, kjer *linspace* določa razpon vrednosti toka in hitrosti na 50 točkah. Nato se ti pari točk združijo v matriko

Xsurf, ki predstavlja kombinacije vhodnih spremenljivk. Funkcija *predict* izračuna napovedi (*Ysurf*) za vse točke na mreži, te pa se nato preoblikujejo v matriko *Zsurf*, da ustreza obliki mreže. Z uporabo funkcije *surf* se izriše 3D-površina napovedanih vrednosti, pri čemer osi x, y in z predstavljajo tok, hitrost in napovedano površino vara. Oznake osi in naslov (*xlabel, xlabel, xlabel, title*) pojasnjujejo vsebino grafa, mreža (*grid on*) pa izboljša berljivost.



Slika 2: Površina napovedi modela GPR za površino vara glede na tok in hitrost podajanja.

Slika 2 prikazuje 3D-površino, ki predstavlja napovedane vrednosti površine vara na podlagi dveh vhodnih spremenljivk: toka (*x*-os, Tok [A]) in podajalne hitrosti (*y*-os, Hitrost [cm/min]). Površina je rezultat Gaussian Process Regression (GPR) modela, naučenega na eksperimentalnih podatkih. Vidna ukrivljenost površine kaže, kako se površina vara nelinearno spreminja glede na kombinacije vhodnih parametrov, kar potrjuje sposobnost zajemanja kompleksnih odvisnosti modela GPR.

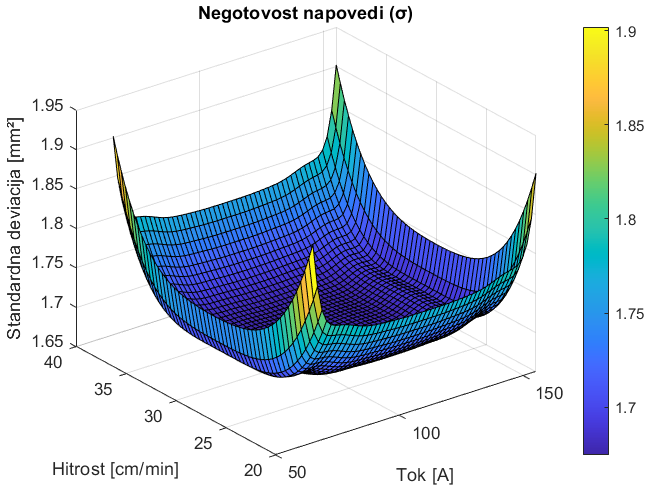
Vizualizacija negotovosti

- Za model GPR prikažite oceno negotovosti napovedi, izražene kot standardna deviacija.

```
% 7. Negotovost
[~, Zstd] = predict(gprMdl, Xsurf);
Zstd = reshape(Zstd, size(tokGrid));

figure;
surf(tokGrid, hitrostGrid, Zstd);
xlabel('Tok [A]');
ylabel('Hitrost [cm/min]');
zlabel('Standardna deviacija [mm²]');
title('Negotovost napovedi (σ)');
colorbar;
grid on;
```

Model s funkcijo *predict* izračuna standardno deviacijo napovedi (*Zstd*) za posamezne točke na vhodni mreži, kar predstavlja negotovost modela. Vrednosti se preoblikujejo v obliko mreže in izrišejo kot 3D-površina, kjer z-os prikazuje negotovost napovedi v mm², barvna lestvica pa ponazarja njeno porazdelitev glede na tok in podajalno hitrost.



Slika 3: Negotovost napovedi modela GPR glede na tok in podajalno hitrost

Slika 3 prikazuje 3D-površino standardne deviacije (σ) napovedi Gaussian Process Regression (GPR) modela. Os X predstavlja tok [A], os Y podajalno hitrost [cm/min], os Z pa standardno deviacijo napovedane površine vara [mm²], ki

označuje negotovost modela v posamezni točki. Najmanjša negotovost je v območju, kjer je model imel več učnih podatkov, medtem ko se proti robovom vrednosti (ekstrapolacijskim območjem) negotovost povečuje, kar je značilno za GPR. Barvna lestvica dodatno poudarja razlike v zanesljivosti napovedi.

2.2 Primer: Nadzorovano učenje za klasifikacijo

Ta primer prikazuje postopek oblikovanja modela za napoved stanja industrijskega stroja na podlagi podatkov, zbranih s senzorji. S pomočjo aplikacije Classification Learner v MATLAB-u boste izdelali binarni klasifikacijski model, ki prepoznava stanje stroja kot "po vzdrževanju" ali "pred vzdrževanjem".

Model boste naučili na podatkih, zbranih tik pred načrtovanim vzdrževanjem in po njem. Predpostavimo, da podatki po vzdrževanju predstavljajo normalno delovanje, medtem ko podatki pred vzdrževanjem nakazujejo nepravilnosti. Tako naučeni model lahko kasneje uporabite za napovedovanje potrebe po vzdrževanju na podlagi novih podatkov.

Podatki za to nalogo so prevzeti iz vira: *Build Condition Model for Industrial Machinery and Manufacturing Processes (www.mathworks.com)*.

2.2.1 Naložite podatke

Primer uporablja nabor podatkov, ki vsebuje 12 značilk, pridobljenih iz triosnih meritev vibracij industrijskega stroja. Izvedite naslednje ukaze za prenos in ekstrahiranje datoteke z naborom podatkov.

- V delovnem prostoru (**Command Window**) izvedite ukaz:

```
url = "https://ssd.mathworks.com/supportfiles/predmaint/" + ...
    "anomalyDetection3axisVibration/v1/vibrationData.zip";

outfilename = websave("vibrationData.zip",url);

unzip(outfilename)
```

Naložite *featureAll* tabelo v datoteko *FeatureEntire.mat*.

➢ V **Command Window** izvedite ukaz:

```
load("FeatureEntire.mat")
```

Tabela vsebuje 17642 meritev za 13 spremenljivk: eno kategorično izhodno spremenljivko (odziv oz. stanje stroja) ter 12 napovednih spremenljivk (značilk).

Tabela 2: Pomen časovnih značilk, ki temeljijo na signalih v časovni domeni.

Značilka	Pomen
RMS	Povprečna energija signala – kaže splošno moč vibracij.
Std (standardni odklon)	Variabilnost signala – koliko se vrednosti odmikajo od povprečja.
Crest Factor	Razmerje med največjo vrednostjo in RMS (kaže prisotnost sunkov ali udarcev).
Kurtosis	Meri šilavost signala (visoka vrednost pomeni ostre impulze).
Skewness	Asimetrija signala (kaže, ali so odkloni bolj v eno smer).
Mean	Povprečna vrednost signala (lahko kaže na premik v osnovni liniji).

Tabela 3: Pomen frekvenčnih značilk, ki so izračunane iz spektra signala.

Značilka	Pomen
SNR (Signal-to-Noise Ratio)	Razmerje med močjo signala in šumom (višja vrednost pomeni bolj čist signal).
THD (Total Harmonic Distortion)	Delež moči harmonikov glede na osnovno frekvenco (kaže na nelinearnosti).
SINAD	Signal + šum + popačenja (celostna ocena kakovosti signala).

Skrajšajte imena značilk tako, da odstranite predpono ("_stats/Col1_").

➢ V **Command Window** izvedite ukaz:

```
for i = 2:13
    featureAll.Properties.VariableNames(i)                          =
erase(featureAll.Properties.VariableNames(i),"_stats/Col1_");
end
```

Predogled prvih osem vrstic tabele.

➢ V **Command Window** izvedite ukaz:

```
head(featureAll)
```

ans=8×13 table

label	ch1CrestFactor	ch1Kurtosis	ch1RMS	ch1Std	ch2Mean	ch2RMS	ch2Skewness	ch2Std	ch3CrestFactor	ch3SINAD	ch3SNR	ch3THD
Before	2.3683	1.927	2.2225	2.2225	-0.015149	0.62512	4.2931	0.62495	5.6569	-5.4476	-4.9977	-4.4608
Before	2.402	1.9206	2.1807	2.1803	-0.018269	0.56773	3.9985	0.56744	8.7481	-12.532	-12.419	-3.2353
Before	2.4157	1.9523	2.1789	2.1788	-0.0063652	0.45646	2.8886	0.45642	8.3111	-12.977	-12.869	-2.9591
Before	2.4595	1.8205	2.14	2.1401	0.0017307	0.41418	2.0635	0.41418	7.2318	-13.566	-13.468	-2.7944
Before	2.2502	1.8609	2.3391	2.339	-0.0081829	0.3694	3.3498	0.36931	6.8134	-13.33	-13.225	-2.7182
Before	2.4211	2.2479	2.1286	2.1285	0.011139	0.36638	1.8602	0.36621	7.4712	-13.324	-13.226	-3.0313
Before	3.3111	4.0304	1.5896	1.5896	-0.0080759	0.47218	2.1132	0.47211	8.2412	-13.85	-13.758	-2.7822
Before	2.2655	2.0656	2.3233	2.3233	-0.0049447	0.37829	2.4936	0.37827	7.6947	-13.781	-13.683	-2.5601

Slika 4: Tabela značilk za vibracijske podatke

Slika 4 prikazuje izpis značilk v ukaznem oknu. Vrednosti v prvem stolpcu so oznake meritev, Before ali After, ki označujejo, ali je meritev opravljena pred vzdrževanjem ali po njem. Preostali stolpci vsebujejo 12 značilk, pridobljenih iz meritev vibracij z uporabo aplikacije *Diagnostic Feature Designer v Predictive Maintenance Toolbox*.

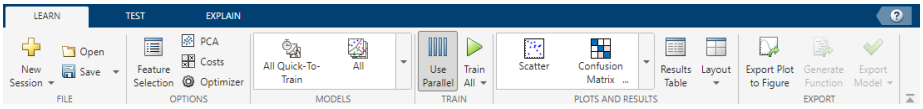
2.2.2 Uvozite podatke v Classification Learner in jih razdelite

Uvozite tabelo featureAll v aplikacijo Classification Learner in pustite 10 % podatkov za testni niz.

➢ Na zavihku **Apps** kliknite puščico **Show more**, da prikažete galerijo aplikacij.
➢ V skupini **Machine Learning and Deep Learning** kliknite **Classification Learner**.

Odpre se novo okno Classification Learner.

➢ Na zavihku **Classification Learner**, v razdelku **File** kliknite **New Session** in **From Workspace**.



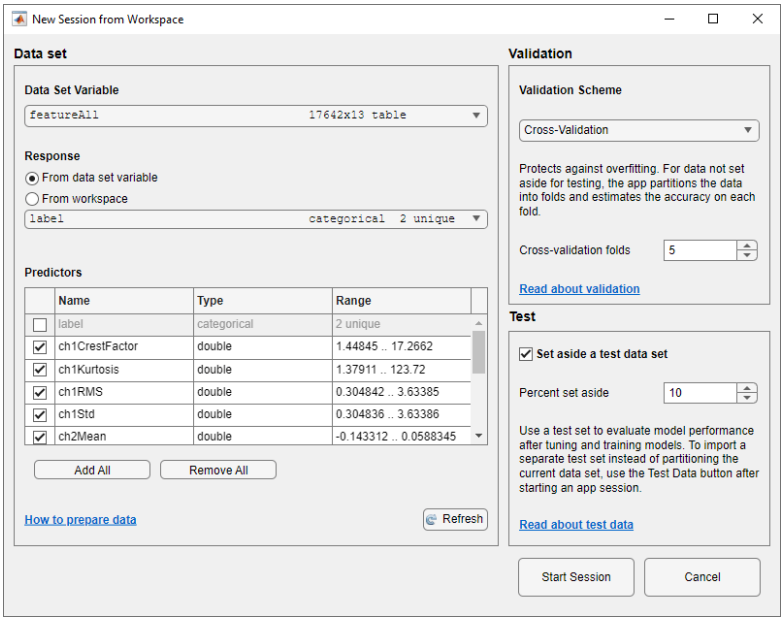
Slika 5: Ukazna vrstica aplikacije Classification Learner

- V oknu **New Session** izberite tabelo **featureAll** na seznamu **Data Set Variable**.

Aplikacija izbere spremenljivke odziva (label) in spremenljivke napovedovanja (Predictors - 12 features) glede na oznake podatkov.

- V razdelku **Test** kliknite potrditveno polje (**set aside a test data set**), da shranite testni niz podatkov.
- Določite **10 %** uvoženih podatkov kot testni niz.

Tabela featureAll vsebuje 17642 vzorcev. Nastavitev 10 % pomeni, da bo 1764 vzorcev v testnem nizu in 15878 vzorcev v učnem nizu.



Slika 6: Razdelitev podatkov v aplikaciji Classification Learner

- Za sprejem sheme preverjanja in nadaljevanje, kliknite **Start Session**.

Privzeta možnost preverjanja veljavnosti je 5-kratna navzkrižna validacija za zaščito pred čezmernim ujemanjem (angl. overfitting). Pri 5-kratni navzkrižni validaciji se podatki razdelijo na pet delov. Model se nato petkrat trenira, vsakič na drugih štirih

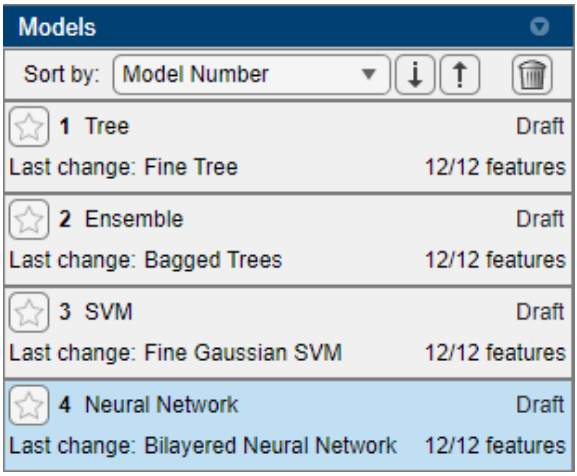
delih podatkov in preveri na preostalem delu. Tako se oceni njegova splošna učinkovitost.

2.2.3 Naučite modele z uporabo vseh značilk

Najprej naučite modele z uporabo vseh 12 značilk. Podokno Models že vsebuje osnutek za fini drevesni model (angl. fine tree model). V podokno Models lahko dodate različne osnutke modelov, tako da jih izberete v galeriji Models.

- Na zavihku **Learn** v razdelku **Models** kliknite puščico **Show more**, da odprete galerijo.
- Izberite tri modele:
  - Bagged trees: V skupini **Ensemble Classifiers** kliknite **Bagged Trees**.
  - Fine Gaussian support vector machine (SVM): V skupini **Support Vector Machines** kliknite **Fine Gaussian SVM**.
  - Bilayered neural network: V skupini **Neural Network Classifiers** kliknite **Bilayered Neural Network**.

Aplikacija Classification Learner vključuje osnutke modelov v podoknu Models.



Slika 7: Izbrani modeli za klasifikacijo

- V razdelku **Train** na zavihku **Learn** kliknite **Train All** in izberite **Train All.**

Classification Learner nauči štiri modele z uporabo vseh 12 značilk.

2.2.4 Ocenite zmogljivost modela

Naučene modele lahko primerjate na podlagi več značilnosti. Ocenite lahko na primer natančnost modela, velikost modela (ki vpliva na potrebe po pomnilniku ali prostoru na disku), računske stroške, povezane z učenjem in testiranjem modela, ter napovedno sposobnost modela.

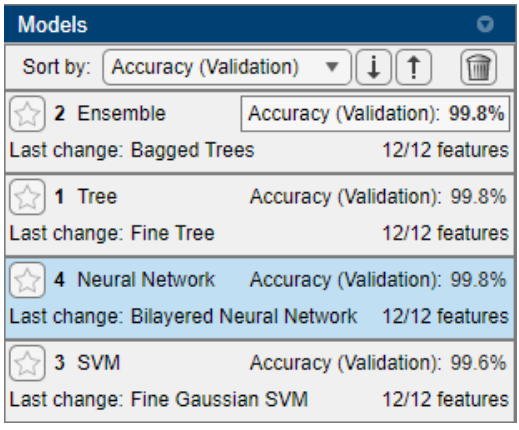
Primerjajte štiri naučene modele glede na natančnost, izmerjeno na podlagi validacijskih podatkov.

V podoknu Models ima vsak model oceno natančnosti preverjanja, ki označuje odstotek pravilno predvidenih odgovorov.

Opomba:

Validacija v rezultate vnese nekaj naključnosti. Vaši rezultati preverjanja modela se lahko razlikujejo od rezultatov, prikazanih v tem primeru.

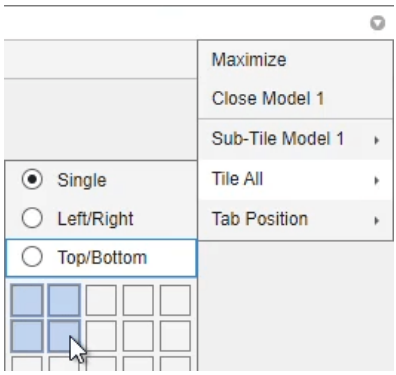
- Razvrstite naučene modele glede na natančnost preverjanja. V podoknu **Models** kliknite puščico **Sort by** in izberite **Accuracy (Validation)**.



Slika 8: Primerjava natančnosti naučenih modelov

Če preverimo tri decimalna mesta, ima model Ensemble največjo natančnost. Za boljše razumevanje rezultatov preuredite postavitev grafov, da boste lahko primerjali matrike zmede (angl. confusion matrix) za štiri modele.

- Kliknite gumb **Document Actions**, ki je skrajno desno od zavihkov izrisa modela.
- Izberite možnost **Tile All** in določite postavitev **2-by-2**.



Slika 9: Postavitev grafov matrike zmede

V zgornjem desnem kotu vsakega izrisa kliknite gumb  , da naredite več prostora za izris.

Slika 10: Primerjava matrik zmede

V primerjavi z drugimi modeli ima model Ensemble (Model 2) najmanj elementov v izvendiagonalnih celicah, ki ustrezajo nepravilno razvrščenim opazovanjem.

2.2.5 Izvozite model v delovni prostor in shranite sejo

Izvozite najboljši model v delovni prostor in preverite velikost modela.

- V podoknu **Models** kliknite model, da ga izberete.
- Na zavihku **Learn** kliknite **Export**, kliknite **Export Model** in izberite **Export Model**.
- Iz izvoženega modela izključite učne podatke tako, da počistite potrditveno polje v ukaznem oknu **Export Classification Model**.

Opomba:

Končni model, ki ga izvozi Classification Learner, se vedno nauči z uporabo celotnega nabora podatkov, razen podatkov, rezerviranih za testiranje. Izbrana shema validacije vpliva le na način, kako aplikacija izračuna metrike zmede modela.

- V ukaznem oknu **Export Classification Model** lahko uredite ime izvožene spremenljivke in nato kliknite **OK**.

V delovnem prostoru se prikaže nova spremenljivka trainedModel. Privzeto ime za izvoženi model je trainedModel in se poveča vsakič, ko izvozite model (na primer trainedModel1), da se prepreči prepisovanje obstoječih izvoženih modelov.

2.2.6 Shranite in zaprite trenutno sejo Classification Learner

- Kliknite **Save** v razdelku **File** na zavihku **Learn**.
- Določite ime in lokacijo datoteke seje in nato zaprite aplikacijo **Classification Learner**.

2.2.7 Preverite velikost modela

Preverite velikost modela z ukazom *whos* v ukaznem oknu.

- V **Command Window** izvedite ukaz:

```
mdl = trainedModel.ClassificationEnsemble;
whos mdl
```

```
  Name      Size              Bytes  Class
Attributes

  mdl       1x1              315622
classreg.learning.classif.CompactClassificationEnsemble
```

Slika 11: Velikost najboljšega naučenega modela

Predpostavimo, da želimo model uporabiti na programirljivem logičnem krmilniku (PLK) z omejenim pomnilnikom in model Ensemble z vsemi 12 značilkami presega vire, ki so na razpolago na PLK-ju.

2.2.8 Nadaljevanje seje Classification Learner

V aplikaciji Classification Learner odprite predhodno shranjeno sejo.

- Kliknite **Open** v razdelku **File**.
- V ukaznem oknu **Select File** izberite shranjeno sejo.

2.2.9 Izberite najpomembnejše značilke z uporabo razvrstitve značilk

Eden od pristopov za zmanjšanje velikosti modela je zmanjšanje števila značilk v modelu z uporabo razvrščanja in izbire značilk. Naučite nove modele z zmanjšanim naborom značilk in ocenite natančnost modela.

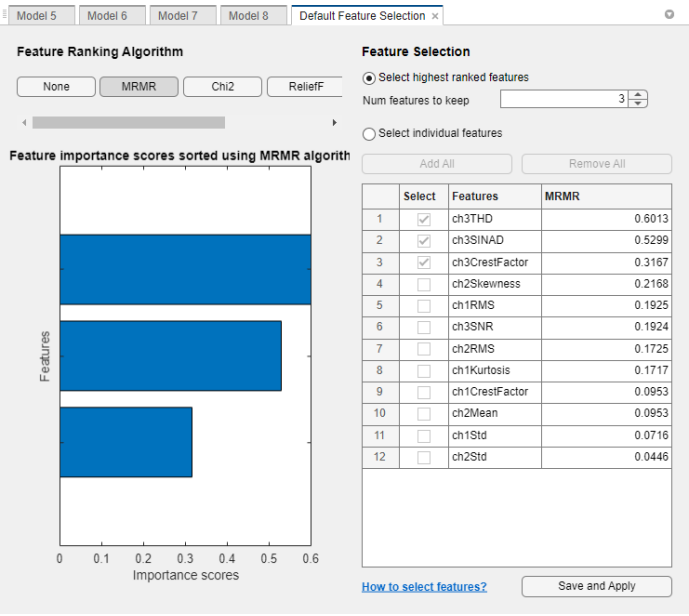
- Ustvarite kopijo vsakega naučenega modela tako, da z desno miškino tipko kliknete model in izberete **Duplicate**.
- Za razvrščanje značilk v **Classification Learner**, kliknite **Feature Selection** v razdelku **Options** na zavihku **Learn**.

Aplikacija odpre zavihek Default Feature Selection.

- Na zavihku **Default Feature Selection** kliknite **MRMR** pod **Feature Ranking Algorithm**.

Aplikacija prikaže palični graf razvrščenih rezultatov po pomembnosti značilk, pri čemer večje ocene (vključno z Infs) kažejo na večjo pomembnost značilke. Tabela na desni prikazuje razvrščene značilke in njihove ocene.

- V razdelku **Feature Selection** uporabite privzeto možnost izbire najvišje uvrščenih značilk.
- Določite, da obdržite **3** značilke za učenje modela.



	Select	Features	MRMR
1	☑	ch3THD	0.6013
2	☑	ch3SINAD	0.5299
3	☑	ch3CrestFactor	0.3167
4	☐	ch2Skewness	0.2168
5	☐	ch1RMS	0.1925
6	☐	ch3SNR	0.1924
7	☐	ch2RMS	0.1725
8	☐	ch1Kurtosis	0.1717
9	☐	ch1CrestFactor	0.0953
10	☐	ch2Mean	0.0953
11	☐	ch1Std	0.0716
12	☐	ch2Std	0.0446

Slika 12: Razvrstitev značilk po pomembnosti

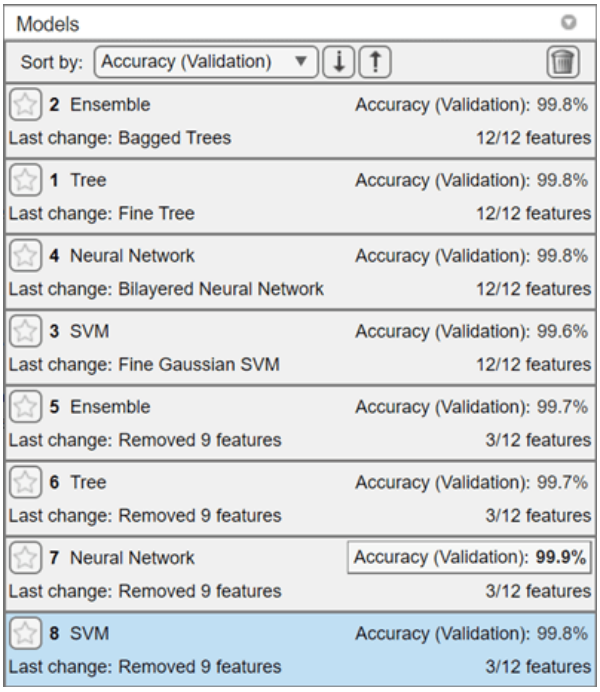
Rezultati razvrščanja značilk temeljijo na celotnem razpoložljivem naboru podatkov, ki vključuje tako podatke za učenje kot podatke za validacijo, ne pa tudi testnih podatkov. Aplikacija pri učenju modela uporabi najvišje uvrščene značilke. Pred vsakim učnim korakom izvede izbiro značilk, nato pa na podlagi izbranih značilk nauči model. Pomembno je poudariti, da lahko pri različnih učnih ponovitvah aplikacija izbere različne značilke kot najpomembnejše, saj se razvrščanje značilk izvaja na novo v vsakem koraku.

- Kliknite **Save and Apply**.

Aplikacija uporabi spremembe izbire značilk za nove osnutke modelov v podoknu Models. Upoštevajte, da osnutki modelov uporabljajo 3 značilke od 12 (3/12).

- V razdelku **Train** kliknite **Train All** in izberite **Train All**.

Aplikacija uči vse nove osnutke modelov z uporabo izbranih značilk.



Slika 13: Natančnost modelov, naučenih na treh najpomembnejših značilkah.

Ugotovimo, da so v tem primeru modeli, ki so bili naučeni z uporabo samo treh značilk, primerljivi z modeli, ki so bili naučeni za vse značilke. Ta rezultat nakazuje, da lahko model, ki temelji samo na treh najboljših značilkah, doseže podobno natančnost kot model, ki temelji na vseh značilkah.

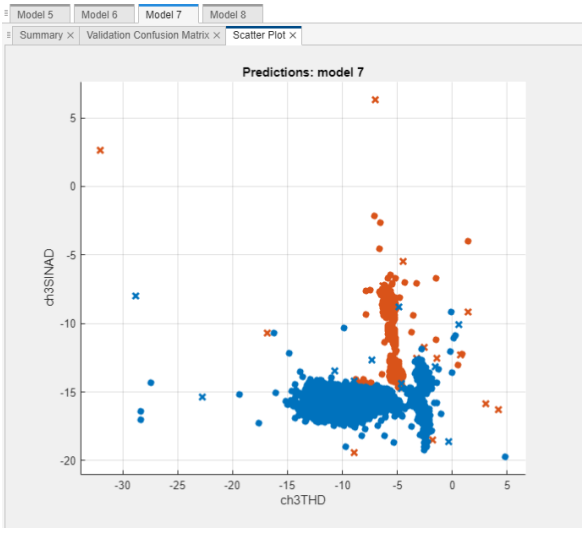
Med vsemi treniranimi modeli je najbolj uspešen model nevronske mreže (Neural Network) s tremi značilkami.

2.2.10 Raziščite pomembne značilke v raztresenem grafikonu

Za vizualno oceno učinkovitosti klasifikacijskega modela si oglejte raztreseni diagram, ki prikazuje razporeditev podatkov glede na dve najpomembnejši napovedni značilki: ch3THD in ch3SINAD. Ker ima izbrani model visoko natančnost, pričakujemo jasno ločene razrede v tem prikazu.

Navodila za prikaz grafa:

- V podoknu **Models** izberite najboljši model. V našem primeru je to Model 7, ki temelji na nevronski mreži.
- Na zavihku **Learn** pojdite v razdelek **Plots and Results** in kliknite **Show more**, da odprete dodatne možnosti.
- V galeriji izberite možnost **Scatter** pod skupino **Validation Results**.
- V razdelku **Predictors** izberite funkciji **ch3THD** (za os *X*) in **ch3SINAD** (za os *Y*).



Slika 14: Raztreseni graf značilk ch3THD in ch3SINAD

Aplikacija nato ustvari raztreseni diagram, kjer so podatkovne točke razporejene glede na napovedi modela. Ker uporabljate navzkrižno preverjanje, model za vsak posamezen primer napovedi ni bil učen na tem konkretnem primeru. To pomeni, da

je vsak izhod rezultat napovedi modela, ki tega podatka ni videl med učenjem, kar zagotavlja realistično in nepristransko oceno učinkovitosti modela.

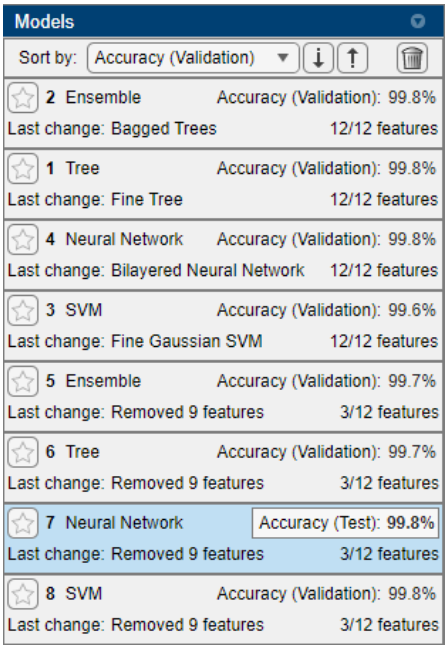
Slika 14 kaže močno ločitev med kategorijama Before in After za obe značilki.

2.2.11 Ocenite natančnost modela na testnem nizu podatkov

Za oceno učinkovitosti modela na novih, še nevidenih podatkih uporabite testni nabor podatkov. V tem primeru boste preverili delovanje modela nevronske mreže.

- V podoknu **Models** izberite model **Neural Network**.
- Na zavihku **Test**, v razdelku **Test**, kliknite **Test Selected**.

Aplikacija izračuna natančnost modela na testnem naboru podatkov, pri čemer uporabi model, ki je bil naučen na celotnem razpoložljivem učnem naboru podatkov. Rezultat potrjuje visoko natančnost modela tudi na neznanih podatkih. Dosežena natančnost na testnem naboru je 99,8 %, kar je primerljivo z validacijskimi rezultati.



Slika 15: Natančnost modela, izračunana na podlagi testnih podatkov.

Prikažite matriko zmede testnega nabora podatkov.

- V razdelku **Plots and Results** na zavihku **Test** kliknite **Confusion Matrix (Test)**.

Slika 16: Matrika zmede za testne podatke

2.2.12 Izvozite končni model

Končni model izvozite v delovni prostor in preverite velikost modela.

- V **Classification Learner** izberite model **Neural Network** v podoknu **Models**.
- V zavihku **Learn** kliknite **Export**, nato kliknite **Export Model** in izberite **Export Model**.
- V ukaznem oknu **Export Classification Model** po potrebi uredite ime izvožene spremenljivke in kliknite **OK**.

Privzeto ime je trainedModel1.

Preverite velikost modela z ukazom *whos*.

- V **Command Window** izvedite ukaz:

```
mdl_final = trainedModel1.ClassificationNeuralNetwork;
whos mdl_final
```

```
  Name           Size            Bytes  Class
  Attributes

    mdl_final      1x1              7842
  classreg.learning.classif.CompactClassificationNeuralNetwork
```

Slika 17: Velikost modela, naučenega na treh značilkah.

Upoštevajte, da je velikost končnega modela (mdl_final iz trainedModel1) manjša od velikosti modela Ensemble (mdl iz trainedModel).

2.3 Samostojno delo s področja nadzorovanega učenja

2.3.1 Izbor in optimizacija klasifikacijskega modela

Cilj: Razviti in oceniti različne klasifikacijske modele za napoved stanja stroja, izbrati najprimernejšega ter raziskati vpliv izbire značilk in kompleksnosti modela.

Navodila:

- Uvozite podatke featureAll v aplikacijo Classification Learner.
- Razdelite podatke z 10 % za testiranje.
- Naučite več modelov (npr. All ali izbrane).
- Primerjajte modele glede na:
  - natančnost (Accuracy),
  - velikost modela (Model size),
  - čas učenja in napovedovanja.
- Za najboljši model:
  - uporabite izbor značilk in ohranite le tri najpomembnejše,
  - ponovno naučite model in preverite uspešnost.
- Prikažite:
  - raztreseni graf za dve najpomembnejši značilki,

- matriko zmede za testne podatke.

➢ Razmislek:
- Kako se spremeni velikost modela pri zmanjšanju števila značilk?
- Ali bi lahko tak model uporabili na napravi z omejenim pomnilnikom (npr. PLK)?
- Kakšen kompromis ste zaznali med kompleksnostjo in natančnostjo?

2.3.2 Samodejno iskanje optimalnih hiperparametrov

Cilj: Uporabiti postopek samodejne optimizacije hiperparametrov za izboljšanje klasifikacijskega modela in primerjati rezultate z modeli s privzetimi nastavitvami.

Hiperparametri so nastavitve modela, ki jih določimo pred učenjem, in vplivajo na proces učenja ter končno delovanje modela (npr. število dreves, hitrost učenja ali širina jedra).

Navodila:

➢ Uvozite podatke featureAll v aplikacijo Classification Learner.
➢ V galeriji modelov izberite enega izmed Optimizable Models (npr. Optimizable Tree, Optimizable SVM ali Optimizable Ensemble).
➢ Kliknite Train in počakajte, da se optimizacija zaključi (algoritem bo samodejno iskal najboljšo kombinacijo hiperparametrov).
➢ Primerjajte rezultate:
- natančnost (angl. Accuracy),
- velikost modela,
- trajanje učenja,
- izbrana konfiguracija (prikazana v razdelku Hyperparameters).

➢ Nastavite ročno izbrane vrednosti hiperparametrov in preverite, ali dobite boljše rezultate.
➢ Razmislek:
- Katere hiperparametre je algoritem optimiziral?
- Ali se je izboljšala natančnost modela v primerjavi s prejšnjo nalogo?

- Bi za podobne naloge v industriji raje uporabili ročno iskanje nastavitev ali avtomatsko optimizacijo? Zakaj?

2.3.3 Primerjava in ocena regresijskih modelov

Cilj: Primerjati različne regresijske modele za napoved površine vara in analizirati njihovo natančnost ter uporabnost za nadaljnje optimizacijske naloge.

Navodila:

- Uvozite podatke iz poglavja 8.1 (Priloga 1) v aplikacijo Regression Learner.
- Naučite vse Quick-to-Train modele.
- Primerjajte modele glede na:
  - RMSE (učna in testna napaka),
  - R^2 (koeficient določnosti).
- Določite model z najboljšim razmerjem med natančnostjo in preprostostjo.
- Za najboljša dva modela izrišite graf:
  - dejanske proti napovedanim vrednostim (y vs. yPred).
- Razmislek:
  - Ali bi izbrani model uporabili za integracijo v optimizacijski algoritem (npr. PSO ali GA)?
  - Ali napake modela nakazujejo sistematična odstopanja (npr. prenaučenost ali slabše prileganje pri robnih vrednostih)?

Univerzitetna založba
Univerze v Mariboru

3 Nenadzorovano učenje

Nenadzorovano učenje (angl. Unsupervised Learning) se uporablja za odkrivanje vzorcev v neoznačenih podatkih. Glavni podskupini sta razvrščanje v skupine oz. gručenje (angl. Clustering) in zmanjševanje dimenzionalnosti (angl. Dimensionality Reduction).

Gručenje je iskanje podobnih skupin v podatkih brez vnaprejšnjih oznak. V inženirstvu se uporablja za razvrščanje delovnih stanj strojev glede na senzorske podatke, odkritje režimov obratovanja v kompleksnih sistemih, analizo porabe energije in segmentacijo odjemalcev, avtomatsko označevanje nestrukturiranih podatkov ipd.

Najpogosteje uporabljene metode gručenja so:

- K-sredinsko gručenje (angl. K-Means): preprost algoritem za razvrščanje podatkov v K skupin z iterativnim posodabljanjem središč, da se minimizira razdalja med podatki in njihovimi skupinami.
- Pomik sredine (angl. Mean-shift): algoritem za razvrščanje podatkov, ki postopno premika središča proti gostejšim regijam podatkov, pri čemer samodejno določi število skupin brez potrebe po vnaprejšnji nastavitvi.

- DBSCAN (angl. Density-Based Spatial Clustering of Applications with Noise): povezuje točke v skupine, kjer je gostota dovolj velika, in učinkovito ločuje šum oziroma izstopajoče točke.
- Hierarhično gručenje (angl. Hierarchical Clustering): algoritem, ki postopno združuje ali razdružuje skupine podatkov in omogoča prikaz strukture na različnih ravneh podrobnosti (granularnosti).
- Gaussian Mixture Models (GMM): verjetnostni algoritem, ki podatke modelira kot mešanico Gaussovih porazdelitev in omogoča mehko (verjetnostno) dodeljevanje točk več skupinam.

Zmanjševanje dimenzionalnosti (angl. Dimensionality Reduction) je metoda za poenostavitev podatkov z ohranjanjem ključnih informacij. V inženirstvu se uporablja za pripravo podatkov za vizualizacijo in razlago modelov, pospešitev učenja z manj vhodnimi spremenljivkami, odpravo redundance v podatkih iz več senzorjev, identifikacijo glavnih vplivnih dejavnikov v procesih (npr. s PCA) ipd.

Najpogosteje uporabljene metode za zmanjševanje dimenzionalnosti so:

- Metoda glavnih komponent (angl. Principal Component Analysis oz. PCA): linearna projekcija največje variance.
- Samoskodirnik (angl. Autoencoders): nevronska mreža, ki se uči stiskati podatke v nižjedimenzionalno predstavitev in jih nato rekonstruirati, s čimer zajame ključne značilnosti vhodnih podatkov.
- t-SNE (angl. t-distributed Stochastic Neighbor Embedding): metoda za nelinearno zmanjševanje dimenzionalnosti, ki omogoča vizualizacijo kompleksnih podatkovnih struktur v 2D- ali 3D-prostoru na način, ki ohranja lokalne odnose med točkami.
- UMAP (angl. Uniform Manifold Approximation and Projection): metoda za zmanjševanje dimenzionalnosti, ki omogoča hitro vizualizacijo podatkov ob ohranjanju tako lokalne kot globalne strukture v podatkovnem prostoru.
- SOM (angl. Self-Organizing Maps): nevronska mreža, ki preslika podatke na nizkodimenzionalno mrežo in pri tem ohranja topološke odnose med podatki.

3.1 Primer: Nenadzorovano učenje za zmanjševanje dimenzionalnosti

V tem primeru bomo uporabili metode nenadzorovanega učenja za vizualizacijo in poenostavitev vibracijskih podatkov industrijskega stroja. S pomočjo metod zmanjševanja dimenzionalnosti bomo iz visokodimenzijskega prostora značilk (12 značilk) poiskali nizkodimenzionalne predstavitve, ki omogočajo vpogled v strukturo podatkov in morebitna stanja stroja. Uporabljeni podatki so pridobljeni z merjenjem vibracij stroja pred vzdrževanjem in po njem.

Cilj je ugotoviti, ali lahko brez uporabe oznak (npr. brez eksplicitne klasifikacije) razpoznamo različna stanja delovanja stroja. Uporabili bomo dve metodi: PCA in t-SNE. Metoda PCA omogoča linearen pogled v prostor največje variance, medtem ko t-SNE omogoča odkrivanje kompleksnejših nelinearnih struktur. Oba pristopa sta ključna koraka pri pripravi podatkov za nadaljnje učenje in razumevanje dinamike industrijskih sistemov.

Podatki za to nalogo so prevzeti iz *www.mathworks.com*.

3.1.1 Naložite podatke

➢ V **Command Window** izvedite ukaz:

```
url = "https://ssd.mathworks.com/supportfiles/predmaint/" + ...
    "anomalyDetection3axisVibration/v1/vibrationData.zip";
outfilename = websave("vibrationData.zip",url);
unzip(outfilename)
```

Naložite *featureAll* tabelo v datoteko *FeatureEntire.mat*.

➢ V ukazni vrstici naložite podatke.

```
load("FeatureEntire.mat");
```

3.1.2 Ustvarite novo datoteko in pripravite podatke

➢ Ustvarite novo datoteko, npr. *grucenje.m*.

➢ Iz tabele **featureAll** izločite numerične značilke in jih standardizirajte.

```
% 1. Izloči numerične značilke
numericVars = varfun(@isnumeric, featureAll, 'OutputFormat',
'uniform');
X = table2array(featureAll(:, numericVars));

% 2. Standardizacija
X_std = (X - mean(X)) ./ std(X);
```

Ta del kode najprej izloči vse numerične značilke iz tabele *featureAll* in jih pretvori v matriko *X*. Nato sledi standardizacija podatkov, pri kateri se vsaki značilki odšteje povprečje in deli s standardnim odklonom, s čimer dobimo podatke s povprečjem 0 in standardnim odklonom 1.

3.1.3 Zmanjševanje dimenzionalnosti z metodo PCA

➢ Izvedite PCA in prikažite podatke v dvorazsežnem prostoru.

```
% 3. PCA
[coeff, score, ~, ~, explained] = pca(X_std);
```

Na standardiziranih podatkih se izvede analiza glavnih komponent (PCA), ki podatke preoblikuje v nov prostor neodvisnih komponent, urejenih po pomembnosti. *coeff* vsebuje smeri glavnih komponent, *score* projekcije podatkov nanje, *explained* pa delež pojasnjene variance vsake komponente.

➢ Dodajte oznako podatkov.

Oznake dodamo zgolj za lažjo interpretacijo rezultatov. PCA kot metoda nenadzorovanega učenja deluje na neoznačenih podatkih, z barvno označitvijo pa lažje presodimo, kako uporabna je redukcija dimenzij za ločevanje različnih stanj.

```
% 4. Poišči stolpec z oznakami (categorical ali string)
isLabelVar = varfun(@(x) iscategorical(x) || isstring(x), featureAll,
'OutputFormat', 'uniform');
labelColIdx = find(isLabelVar, 1);
```

```
if isempty(labelColIdx)
    error("V tabeli ni stolpca z oznakami (npr. 'Condition' ali
'Label').");
end

labels = featureAll{:, labelColIdx};  % izbere ustrezen stolpec
```

Ta del kode poišče stolpec z oznakami v tabeli *featureAll*, tako da preveri, kateri stolpci so tipa kategorični ali znakovni. Če tak stolpec obstaja, se njegov indeks shrani v *labelColIdx* in iz njega pridobijo oznake *labels*. Če ga ni, se sproži napaka. Oznake se nato uporabijo za barvanje točk pri vizualizaciji ali za učenje modelov.

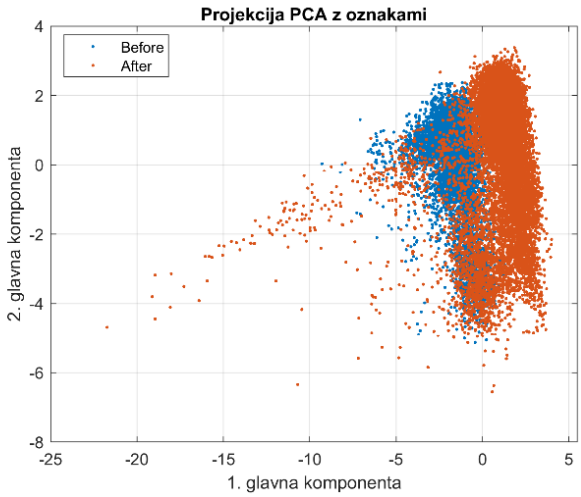
3.1.4 Analiza rezultatov

- Prikažite rezultate PCA v dvorazsežnem prostoru.

```
% 5. Vizualizacija PCA z oznakami
figure;
gscatter(score(:,1), score(:,2), labels);
xlabel('1. glavna komponenta');
ylabel('2. glavna komponenta');
title('Projekcija PCA z oznakami');
grid on;
```

Ta del kode izriše dvodimenzionalno projekcijo podatkov v prostoru prvih dveh glavnih komponent, pri čemer so točke obarvane glede na pripadajoče oznake. Funkcija *gscatter* nariše raztreseni diagram z barvami po skupinah, kar omogoča vizualno analizo ločenosti med razredi po izvedbi PCA.

Slika 18 prikazuje rezultat zmanjševanja dimenzionalnosti z metodo glavnih komponent (PCA), kjer so vibracijski podatki industrijskega stroja projicirani v dvorazsežni prostor. Točke so označene glede na stanje stroja: Before (modro) pomeni pred vzdrževanjem, After (rdeče) pa po vzdrževanju. PCA ne uporablja oznak pri učenju, barvna oznaka je dodana zgolj za lažjo interpretacijo. Opazimo lahko delno ločenost med obema stanjema, kar nakazuje, da značilke vsebujejo uporabne informacije za razlikovanje delovanja stroja pred vzdrževanjem in po njem.



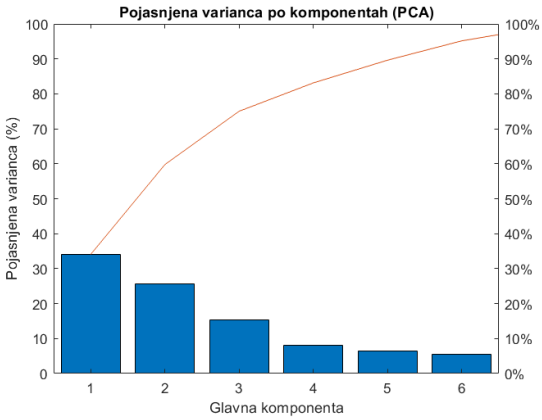
Slika 18: Vizualizacija podatkov z metodo PCA in barvnimi oznakami stanja stroja

- Varianco posameznih komponent predstavite s Pareto diagramom.

```
% 6. Pareto graf variance
figure;
pareto(explained);
xlabel('Glavna komponenta');
ylabel('Pojasnjena varianca (%)');
title('Pojasnjena varianca po komponentah (PCA)');
```

Ta del kode izriše Pareto diagram, ki prikazuje, kolikšen delež celotne variance pojasni vsaka glavna komponenta. Os X predstavlja posamezne komponente, os Y pa delež pojasnjene variance v odstotkih. Ta vizualizacija pomaga določiti, koliko komponent je smiselno ohraniti za nadaljnjo analizo.

Slika 19 prikazuje Pareto diagram pojasnjene variance po glavnih komponentah metode PCA. Stolpci predstavljajo delež variance, ki jo pojasni vsaka posamezna komponenta, medtem ko rdeča črta prikazuje kumulativno pojasnjeno varianco. Vidimo, da prvi dve komponenti skupaj pojasnita več kot 60 % variance v podatkih, kar upravičuje njuno uporabo za vizualizacijo in analizo. Prispevek dodatnih komponent je postopno manjši, kar pomeni, da lahko večino informacij ohranimo že z nizkim številom dimenzij.



Slika 19: Prispevek posameznih komponent k pojasnjeni varianci pri PCA

3.1.5 Zmanjševanje dimenzionalnosti z metodo t-SNE

➢ Izvedite t-SNE za nelinearno dimenzionalno zmanjšanje.

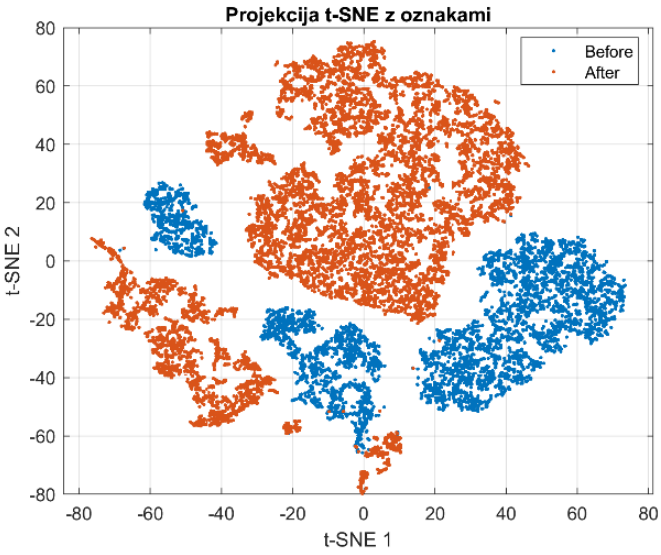
```
% 7. t-SNE za nelinearno zmanjšanje (z oznakami)
Y_tsne = tsne(X_std, 'NumDimensions', 2, 'Perplexity', 30);
```

Ta ukaz izvede t-SNE (angl. t-distributed Stochastic Neighbor Embedding), metodo za nelinearno zmanjševanje dimenzionalnosti, ki podatke iz visokodimenzionalnega prostora preslika v 2D-prostor. Uporablja standardizirane podatke *X_std*, nastavljena pa sta dimenzionalnost ciljne projekcije (*NumDimensions* na 2) in parameter *Perplexity* na 30, ki uravnava ravnotežje med lokalno in globalno strukturo. Rezultat *Y_tsne* vsebuje koordinate podatkov v 2D-prostoru za vizualizacijo.

➢ Prikažite rezultate t-SNE v dvorazsežnem prostoru.

```
% 8. Vizualizacija t-SNE z oznakami
figure;
gscatter(Y_tsne(:,1), Y_tsne(:,2), labels);
xlabel('t-SNE 1');
ylabel('t-SNE 2');
title('Projekcija t-SNE z oznakami');
grid on;
```

Ta del kode izriše 2D-vizualizacijo rezultatov t-SNE, kjer so točke v prostoru t-SNE prikazane glede na prvi in drugi koordinatni stolpec iz *Y_tsne*. Funkcija *gscatter* poskrbi za barvanje točk glede na pripadajoče oznake *labels*, kar omogoča pregled nad razporeditvijo in ločljivostjo skupin po nelinearni projekciji.



Slika 20: Projekcija t-SNE vibracijskih podatkov z oznakami stanja stroja

Slika 20 prikazuje rezultat zmanjševanja dimenzionalnosti s pomočjo metode t-SNE. Podatki so projicirani v dvorazsežni prostor, kjer vsaka točka predstavlja en vzorec vibracij, barva pa označuje stanje stroja pred vzdrževanjem in po vzdrževanju. t-SNE kot nenadzorovana metoda ne uporablja oznak pri projekciji, barvna oznaka omogoča le lažjo interpretacijo rezultatov. Jasna ločenost med skupinama kaže, da značilke vsebujejo zadostno informacijo za razločevanje stanj, obenem pa se znotraj skupin nakazujejo podstrukture, kar kaže na raznolikost režimov delovanja ali stopnje obrabe.

3.2 Primer: Nenadzorovano učenje za gručenje

Ta primer uporabi podatke obrabe orodja, pridobljene po struženju različnih materialov z različnimi parametri obdelave. Podatke boste uvozili v aplikacijo za nenadzorovano učenje z namenom, da prepoznate skupine (gruče) podatkov s

podobnimi lastnosti, tj. vrednostmi merjenih parametrov. Podatki bodo neoznačeni, kar pomeni, da ne bomo ločevali med odvisnimi in neodvisnimi spremenljivkami.

Cilj naloge je ugotoviti, ali algoritem podatke združi v skupino z veliko obrabo in v skupino z zmerno obrabo orodja brez predhodne informacije.

3.2.1 Ustvarite novo datoteko

- Kliknite **New** in izberite **Live Script**.

Live Script v MATLAB-u je interaktivni skript, ki omogoča kombinacijo kode, razlag, enačb in grafov v enem dokumentu. Uporablja se za bolj pregledno analizo, učenje in poročanje rezultatov.

- Kliknite **Save** in **Save As**.
- Vnesite ime datoteke (npr. *grucenje.mlx*) in kliknite **Save**.

3.2.2 Naložite podatke

Podatki za to nalogo so na voljo v poglavju 8.2 (Priloga 2). Na podlagi podatkov ustvarite datoteko CSV z imenom *obraba_orodja.csv*.

- Povlecite in spustite datoteko *obraba_orodja.csv* v polje **Files** in počakajte, da se podatki prenesejo.
- Kliknite **OK**.
- V dokument *grucenje.mlx* vnesite naslednje ukaze.

```
rng(1);
A =
readmatrix('obraba_orodja.csv','Range','A2:E109','DecimalSeparator','
,');
```

Ukaz *rng(1)* nastavi začetno vrednost generatorja naključnih števil, kar omogoča ponovljivost rezultatov pri algoritmih, ki vključujejo naključne elemente. Nato se z ukazom *readmatrix* iz datoteke *obraba_orodja.csv* preberejo podatki v območju od celice *A2* do *F109*, pri čemer je decimalno ločilo vejica. Preskoči se prva vrstica, ki vsebuje

naslove stolpcev, in se preberejo samo numerični podatki. Rezultat je matrika *A*, ki vsebuje eksperimentalno matriko in jo bomo v nadaljevanju uporabili kot vhod v algoritem gručenja.

- V zavihku **Editor** v skupini **Run** kliknite **Run**.

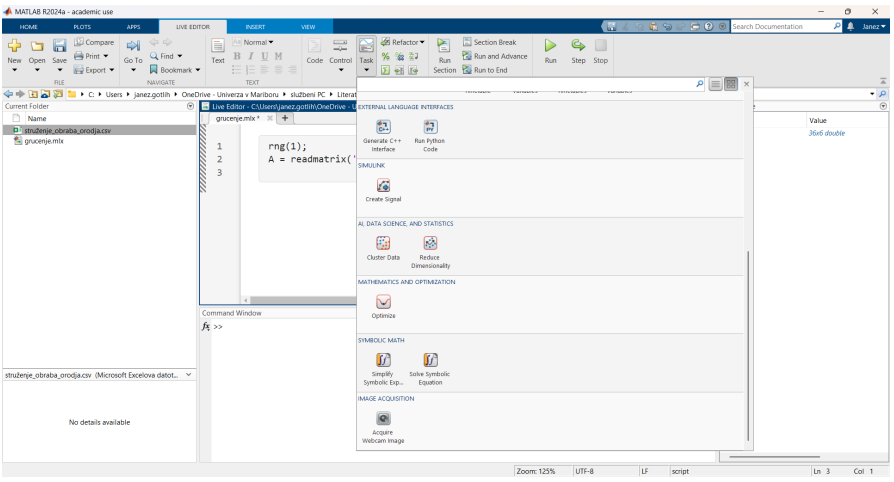
V delovnem prostoru se ustvari matrika *A* dimenzije *108 x 5* tipa *double*.

Dodajte Cluster Data

- V zavihku **Live Editor** v skupini **CODE** kliknite puščico pod ukazom **Tasks**.

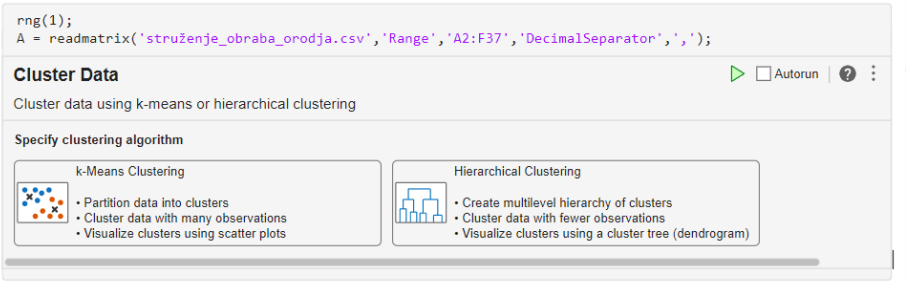
Tasks v MATLAB Live Scriptu so interaktivna orodja, ki omogočajo izvajanje analiz brez programiranja, pri čemer se koda samodejno generira glede na izbrane parametre.

- V skupini **AI, Data Science, and Statistics** kliknite **Cluster Data**.



Slika 21: Dodajanje interaktivne aplikacije Cluster Data za nenadzorovano učenje

V Live Script se vključi interaktivni programski vmesnik za nenadzorovano učenje.



Slika 22: Interaktivni programski vmesnik za nenadzorovano učenje

3.2.3 Združite podatke v dve skupini

- V **Cluster Data** v polju **Specify clustering algorithm** izberite **Hierarhical Clustering**.

Hierarhical Clustering je primernejši za gručenje, ko imamo manjše število vhodnih podatkov.

- Za **Select** data za **Input data** izberite **A**.
- V polju **Specify number of clusters** za **Selection method** izberite **Manual num clusters**, za **Max number of clusters** pa izberite **2**.

S tem določimo matriko vhodnih podatkov in največje število gruč, ki jih želimo prepoznati.

- V polju **Specify optional distance parameters** za **Distance** izberite **Euclidean** in za **Linkeage** izberite **Average**.

Za način izračuna matrike parnih razdalj med podatkovnimi točkami v matriki *A* določimo evklidsko razdaljo. Z ukazom *linkage* in argumentom *average* določimo metodo hierarhično povezovanje podatkov. Z izbrano metodo povprečne povezave se razdalja med gručama izračuna kot povprečje razdalj med vsemi pari točk iz obeh gruč.

- V polju **Display results** izberite **2D-scatter plot** in za **Columns** izberite **1, 2**.

Za grafični prikaz rezultatov uporabimo 2D-raztreseni diagram.

➢ V zavihku **Editor** v skupini **Run** kliknite **Run**.

Opomba:

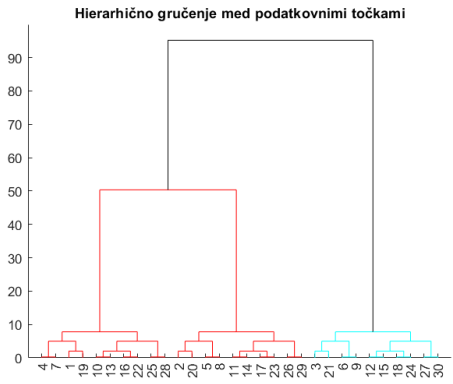
Z razširitvijo spodnjega odseka »Show code« si lahko ogledate programsko kodo za izvedbo gručenja, ki temelji na izbranih parametrih. Koda omogoča naprednejšo analizo kot uporabniško prijaznejši grafični vmesnik, saj omogoča večjo prilagodljivost in transparentnost pri izvajanju postopkov gručenja.

3.2.4 Preglejte rezultate gručenja

Po izvedbi kode se v delovni prostor zapiše vektor *clusterindices*. To je vektor celih števil, dolg toliko, kot je število podatkovnih točk (vrstice v podatkovni matriki A). Vsak element v tem vektorju označuje, v katero gručo je bila uvrščena posamezna točka.

➢ V delovnem prostoru dvokliknite **clusterindices**.
Primerjava clusterindices s podatki iz tabele *struženje_obraba_orodja.csv* pokaže, da so posebna skupina samo primeri s hitrostjo rezanja 220 m/min.

➢ Preverite dendrogram.

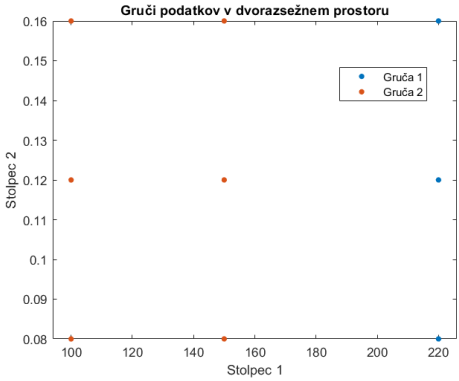


Slika 23: Hierarhično gručenje med podatkovnimi točkami

Slika 23 prikazuje dendrogram, tj. rezultat hierarhičnega gručenja. Vsaka veja predstavlja povezovanje posameznih točk ali gruč glede na njihovo medsebojno podobnost (po evklidski razdalji). Navpična os prikazuje razdaljo med gručami ob združitvi. Nižja kot je povezava, bolj podobni so podatki. Vodoravna os prikazuje identifikatorje podatkovnih točk, torej vrstice v vhodni podatkovni matriki. Na osnovi izbranega praga smo dobili dve gruči.

➢ Preverite raztreseni grafikon.

Slika 24 prikazuje dvorazsežni raztreseni diagram gručenih podatkov. Na podlagi prikaza potrdite, da so podatki združeni po rezalni hitrosti 220 m/min.



Slika 24: Vizualizacija gručenih podatkov v dvorazsežnem prostoru

3.2.5 Dodajte podatke velike obrabe orodja

Podatki v poglavju 8.2 (Priloga 2), označeni z oznako velika obraba orodja, predstavljajo primere, kjer je do večje obrabe prišlo zaradi velike globine rezanja.

➢ Spremenljivko **A** prepišite z novimi podatki z ukazom:

```
A =
readmatrix('obraba_orodja.csv','Range','A2:D163','DecimalSeparator','
,');
```

- V **Cluster Data** v polju **Display results** izberite **3D-scatter plot** in za **Columns** izberite **1, 2, 3**.
- V zavihku **Editor** v skupini **Run** kliknite **Run**.

3.2.6 Preglejte rezultate gručenja

- V delovnem prostoru dvokliknite **clusterindices**.

Primerjava clusterindices s podatki iz tabele *obraba_orodja.csv* pokaže, da so posebna gruča podatki z rezalno hitrostjo 220 m/min.

- Preverite raztreseni grafikon.

Slika 25: 3D-prikaz gručenja z uporabo umetno ustvarjenih podatkov

Potrdite, da so posebna skupina samo podatki z rezalno hitrostjo 220 m/min.

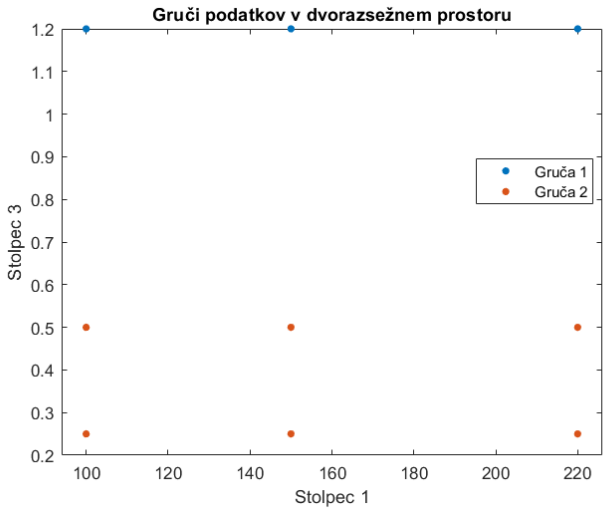
3.2.7 Prilagodite parametre gručenja

- V **Cluster Data** v **Specify optional distance parameters** za **Distance** izberite **Standardized Euclidean** in za **Linkeage** izberite **Average**.

S tem ste spremenili merilo za gručenje podatkov, tako da se uporablja standardizirana različica evklidske razdalje, kjer se vsak vhodni parameter najprej normalizira glede na svojo varianco. Merilo uporabimo, kadar imajo različni vhodni podatki različne merske enote ali različne razpone vrednosti (npr. rezalna hitrost v m/min. in čas obdelave v min.).

- V polju **Display results** izberite **2D-scatter plot** in za **Columns** izberite **1, 3**.
- V zavihku **Editor** v skupini **Run** kliknite **Run**.

Prvi stolpec v matriki A je rezalna hitrost, tretji stolpec v matriki A pa je globina rezanja.



Slika 26: Pravilno združeni umetno ustvarjeni podatki

Zdaj so podatki združeni po globini rezanja, ki najbolj izrazito vpliva na obrabo orodja. Algoritem s temi nastavitvami je uspešno prepoznal podatke z veliko globino rezanja kot ločeno gručo.

3.3 Samostojno delo s področja nenadzorovanega učenja

3.3.1 Primerjava metod gručenja

Cilj: Preučiti, kako se rezultati gručenja spreminjajo glede na izbrano metodo.

Navodila:

- Uporabite podatke iz poglavja 3.2.
- Primerjajte rezultate za vsaj tri metode gručenja: K-Means, Hierarhično, DBSCAN.
- Uporabite različne metrične razdalje (npr. Euclidean, Standardized Euclidean).
- Vizualizirajte rezultate in komentirajte, katera metoda najbolje loči podatke z veliko obrabo orodja.

3.3.2 Uporaba metode silhouette za oceno gruč

Cilj: Kvantitativna ocena kakovosti gručenja.

Navodila:

- Uporabite podatke iz poglavja 3.2.
- Izračunajte silhouette coefficient za rešitve z 2–5 gručami.
- Na podlagi vrednosti predlagajte optimalno število gruč.
- Uporabite metode K-Means in Hierarchical.
- Prikažite silhouette graf.

Opomba:

Za oceno kakovosti gručenja lahko uporabite silhouette coefficient (koeficient obrisa), ki meri, kako dobro je posamezen vzorec združen s točkami v svoji gruči v primerjavi z drugimi gručami. Vrednosti so med –1 in 1, pri čemer višje vrednosti pomenijo boljše gručenje.

V MATLAB-u lahko uporabite funkcijo silhouette(X, idx) za grafični prikaz ali evalclusters(X, 'kmeans', 'silhouette', 'KList' ...) za samodejno izbiro optimalnega števila gruč.

3.3.3 Gručenje na osnovi projekcije PCA ali t-SNE

Cilj: Preveriti uporabnost nizkodimenzionalne projekcije kot osnove za gručenje.

Navodila:

- Uporabite podatke iz poglavja 3.2.
- Zmanjšajte dimenzionalnost z metodo PCA (ali t-SNE).
- Izvedite gručenje v 2D-prostoru (npr. K-Means na komponentah 1 in 2).
- Primerjajte rezultate gručenja z rezultati na originalnih podatkih.

3.3.4 Razširitev spremenljivk in njihova vplivnost

Cilj: Oceniti vpliv posameznih vhodnih spremenljivk na rezultat gručenja.

Navodila:

- Uporabite podatke iz poglavja 3.2.
- Iz podatkov odstranite po eno spremenljivko naenkrat in opazujte spremembo rezultatov gručenja (npr. ali še vedno loči primere z veliko obrabo).
- Opišite, kateri atributi so najpomembnejši za ločevanje gruč.

3.3.5 Vizualizacija gruč z oznakami

Cilj: Prikazati gruče v 2D- ali 3D-prostoru z barvnimi oznakami zmerna/velika obraba zgolj za vizualno interpretacijo.

Navodila:

- Uporabite podatke iz poglavja 3.2.
- Po izvedenem gručenju prikažite gruče v 2D- ali 3D-prostoru z barvnimi oznakami.
- Ocenite, ali je gručenje skladno z dejanskimi stanji.

Univerzitetna založba
Univerze v Mariboru

4 Učenje z okrepitvijo

Učenje z okrepitvijo (angl. Reinforcement Learning) se uporablja za optimizacijo procesov in avtomatizacijo odločanja. Model se uči preko povratnih informacij iz okolja (nagrade/kazni) in išče optimalno zaporedje dejanj. V inženirstvu se uporablja za prilagajanje krmilnih strategij v realnem času, optimizacijo proizvodnih procesov brez vnaprej znanega modela, pametni nadzor energetsko intenzivnih sistemov, terminiranje in razporejanje v delavniških proizvodnjah, samodejno vodenje mobilnih robotov (npr. AGV, droni) ipd.

Najpogosteje uporabljene metode učenja z okrepitvijo so:

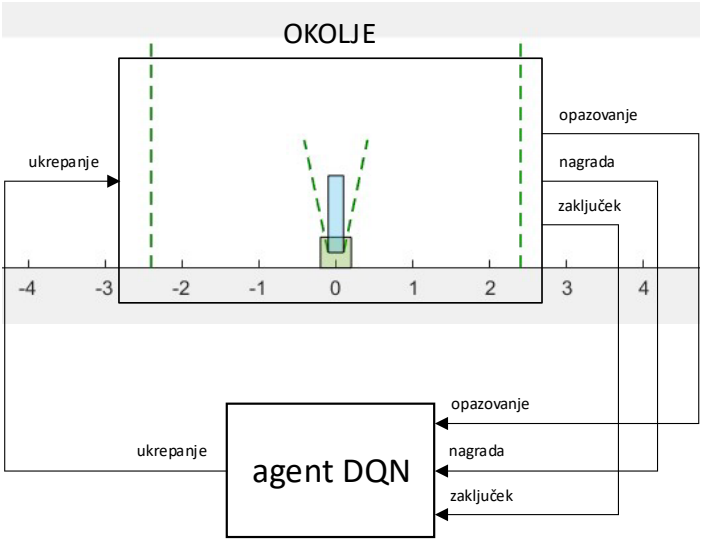
- Q-učenje (angl. Q-learning): z uporabo Q-tabele ocenjuje koristnost posameznih dejanj v določenih stanjih na podlagi prejetih nagrad iz okolja.
- R-učenje (angl. R-learning): različica Q-learninga, prilagojena za okolja z nenehnimi ali konstantnimi nagradami, kjer je cilj maksimirati dolgoročno povprečje nagrad namesto utežene vsote.
- TD-učenje (anlg. Temporal Difference learning): metoda, ki ocenjuje vrednost stanj ali dejanj na podlagi razlike med pričakovano in dejansko nagrado, brez potrebe po končani epizodi.

- Globoke Q-mreže (angl. Deep Q-Networks DQN): razširitev Q-learninga, kjer se Q-vrednosti aproksimirajo z globokimi nevronskimi mrežami, kar omogoča uporabo v kompleksnih okoljih z veliko stanji.
- Metoda gradienta politike (angl. Policy Gradient): metoda, ki ne ocenjuje vrednosti stanj ali dejanj, temveč neposredno optimizira politiko (strategijo izbiranja dejanj) z uporabo gradientnih metod.
- Proksimalna optimizacija politik (anlg. Proximal Policy Optimization PPO): izboljšana metoda učenja politike, ki s posebnimi mehanizmi omeji spremembe strategije med učenjem ter s tem zagotavlja večjo stabilnost in zanesljivost učenja v neznanem okolju.

4.1 Primer: Učenje z okrepitvijo

Ta primer prikazuje, kako oblikovati in usposobiti agenta DQN (deep Q-learning network) z uporabo diskretnega akcijskega prostora v Matlab aplikaciji Reinforcement Learning Designer.

Cilj naloge je agenta DQN naučiti uravnotežiti palico na premikajočem se vozičku z uporabo vodoravnih sil na voziček (Slika 27).



Slika 27: Koncept delovanja učenja z okrepitvijo za primer balansiranja palice

Okolje je definirano z naslednjimi robnimi pogoji:

- Pokončni uravnoteženi položaj palice je 0 radianov, vodoravni položaj palice pa je pi radianov.
- Začetni pokončni kot palce je med –0,05 in 0,05 radiana.
- Zunanja sila na voziček je lahko –10 ali 10 N.
- Opazovanja iz okolja so položaj in hitrost vozička ter kot nagiba in kotna hitrost palice.
- Učna epizoda se konča, če je palica nagnjena za več kot 12 stopinj od navpičnice ali če se voziček premakne za več kot 2,4 m od prvotnega položaja.
- Nagrada +1 je zagotovljena za vsak diskretni časovni korak, ko palica ostane pokonci. Ko palica pade, se uporabi kazen –5.

Naloga je za skripto prirejena po viru: *Design and Train Agent Using Reinforcement Learning Designer (www.mathworks.com)*.

4.1.1 Odprite aplikacijo Reinforcement Learning Designer

- V **Command Window** vpišite in izvedite ukaz:

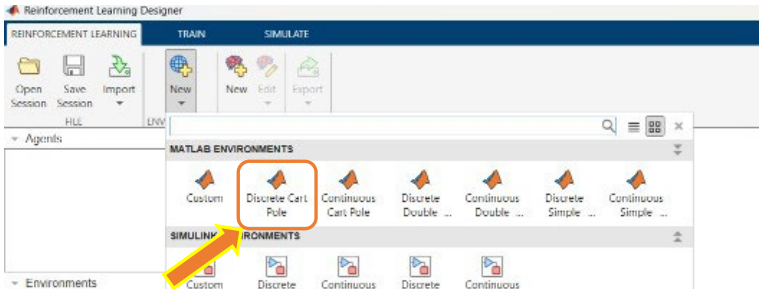
```
reinforcementLearningDesigner
```

Odpre se novo okno Reinforcement Learning Designer.

4.1.2 Uvozite okolje Cart-Pole

Za ta primer uporabite vnaprej določeno diskretno okolje Cart-Pole (voziček-palica).

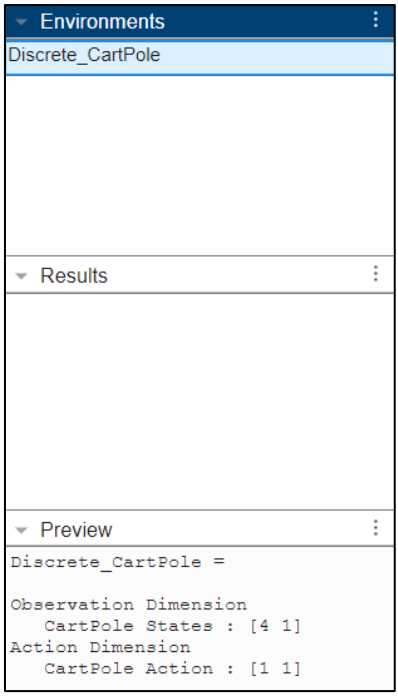
- Na zavihku **Reinforcement Learning** v razdelku **Environments** izberite **New** > **Discrete Cart-Pole** (Slika 28).



Slika 28: Testno okolje za balansiranje palice na vozičku (Discrete Cart Pole)

V podoknu Environment se v aplikaciji doda uvoženo okolje Discrete Cart Pole. Če želite preimenovati okolje, kliknite besedilo okolja. V sejo lahko uvozite tudi več okolij.

Če si želite ogledati dimenzije prostora za opazovanje in delovanje, kliknite ime okolja. Aplikacija prikaže dimenzije v podoknu Preview (Slika 29).



Slika 29: Okolje, včitano v Reinforcement Learning Designer.

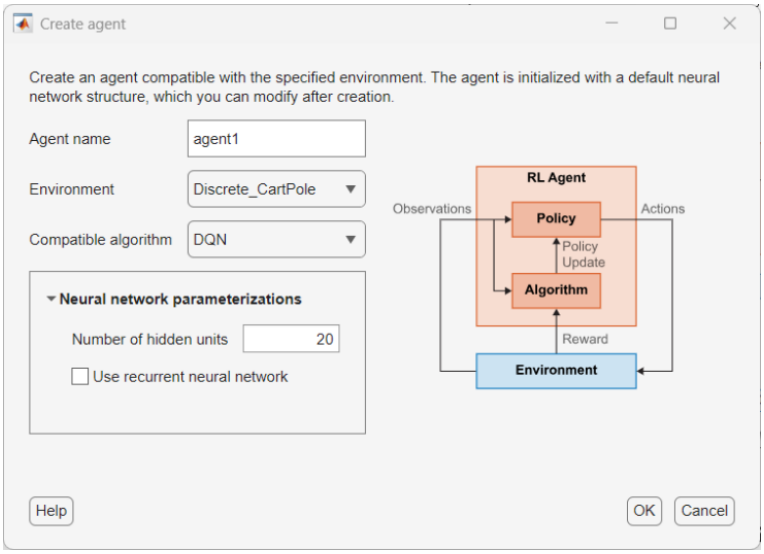
To okolje ima zvezen štiridimenzionalni prostor za opazovanje (položaji in hitrosti vozička in nagibi in kotne hitrosti palice) in diskretni enodimenzionalni akcijski prostor, sestavljen iz dveh možnih sil, –10 N ali 10 N.

4.1.3 Ustvarite agenta DQN za uvoženo okolje

➢ Na zavihku **Reinforcement Learning** v razdelku **Agent** kliknite **New**.

V ukaznem oknu Create agent (Slika 30) podajte ime agenta, okolje in algoritem za učenje. Privzeta konfiguracija agenta uporablja uvoženo okolje in algoritem DQN.

➢ Za ta primer spremenite **Number of hidden units** iz **256** na **20**.

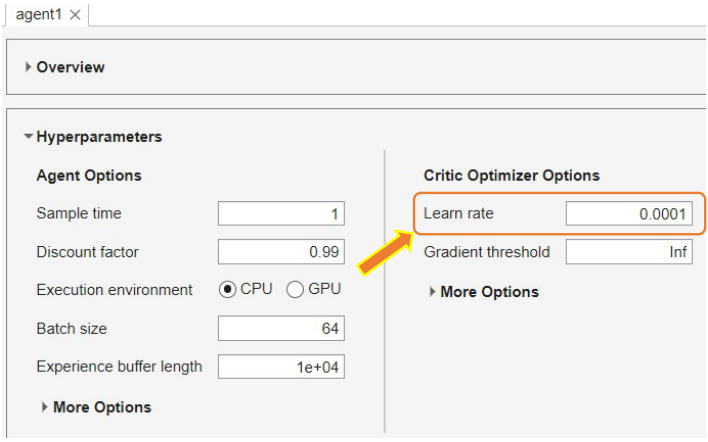


Slika 30: Ustvarjanje agenta za učenje z okrepitvijo

➢ Kliknite **OK**.

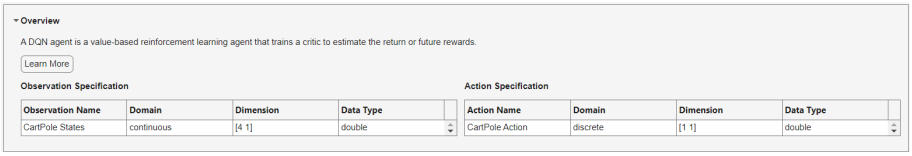
Aplikacija doda novega agenta v podokno Agents in odpre ustrezen dokument agent1.

- V razdelku **Hyperparameter** pod možnostmi **Critic Optimizer Options** nastavite stopnjo učenja **Learn rate** na **0,0001** (Slika 31).



Slika 31: Nastavitve hiperparametrov agenta

- Za povzetek nastavitev agenta DQN ter za ogled specifikacij za opazovanje in ukrepanje za agenta razširite **Overview** (Slika 32).

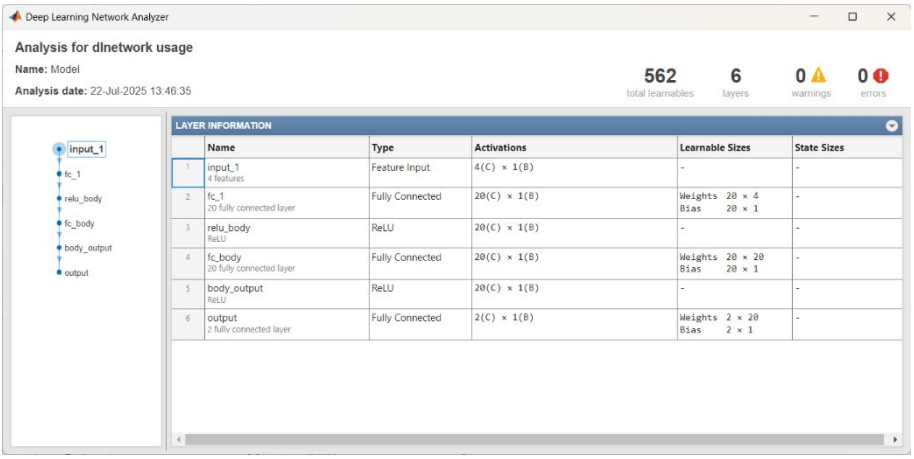


Slika 32: Povzetek nastavitev agenta in okolja

Ko ustvarite agenta DQN v Reinforcement Learning Designerju, agent za svojega kritika uporabi privzeto strukturo globoke nevronske mreže.

- Da si ogledate mrežo, v ukazni vrstici na zavihku **DQN Agent** kliknite **View Critic Model**.

Odpre se okno Deep Learning Network Analyzer in prikaže strukturo mreže.



Slika 33: Struktura globoke nevronske mreže za učenje z okrpitvijo

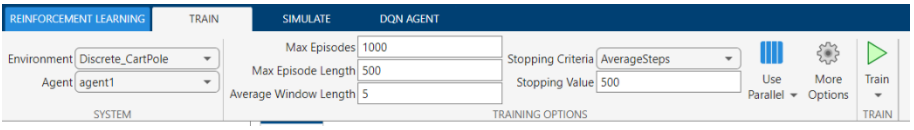
➢ Zaprite **Deep Learning Network Analyzer**.

4.1.4 Naučite Agenta

Da naučite agenta, določite možnosti za učenje agenta.

➢ Na zavihku **Train** določite največje število epizod učenja tako, da nastavite **Max Episodes** na **1000**.

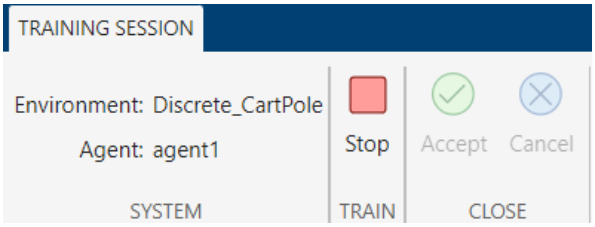
Za druge možnosti učenja uporabite privzete vrednosti. Privzeto merilo za ustavitev je takrat, ko je povprečno število korakov na epizodo (v zadnjih 5 epizodah) večje od 500.



Slika 34: Nastavitve možnosti učenja

➢ Za začetek učenja kliknite **Train**.

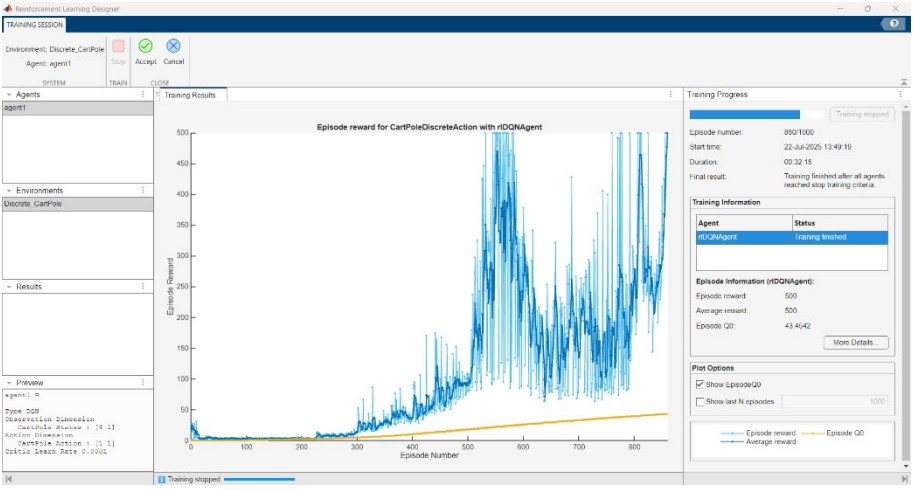
Med učenjem aplikacija odpre zavihek Training Session in prikaže napredek učenja v dokumentu Training Results. Med učenjem lahko kliknete gumb Stop ali Stop Training, da prekinete učenje in izvedete druge operacije v ukazni vrstici (Slika 35).



Slika 35: Možnosti med učenjem

Resume, Accept in Cancel v zavihku Training Session nudijo možnost, da nadaljujete učenje, sprejmete rezultate učenja ter shranite rezultate učenja in naučenega agenta ali prekličete učenje.

- Za nadaljevanje učenja kliknite **Resume**.



Slika 36: Potek učenja z DQN-agentom

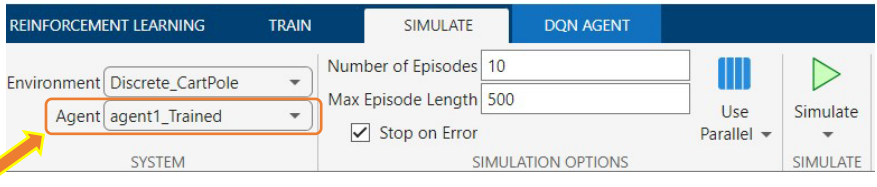
Učenje se ustavi, ko je povprečno število korakov (akumulirana nagrada) na epizodo 500 (Slika 36).

- Kliknite **Accept**, da sprejmete rezultate učenja.

V podoknu Agents se v aplikaciji doda naučeni agent, agent1_Trained.

4.1.5 Simulirajte agenta in preglejte rezultate simulacije

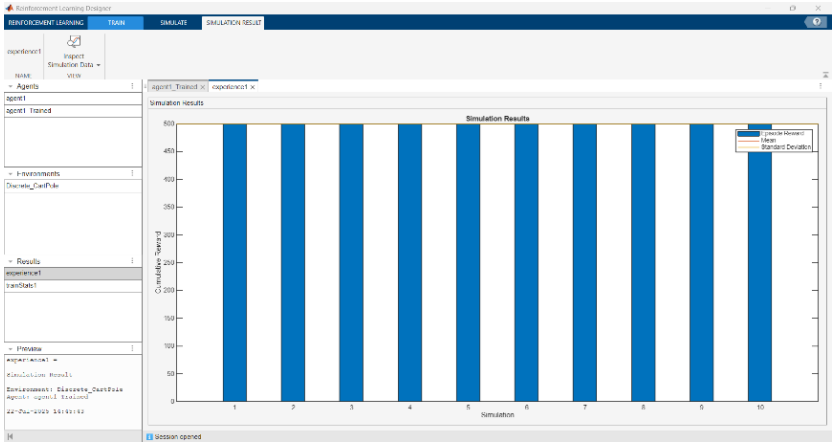
- Na zavihku **Simulate** izberite **agent1_Trained** na spustnem seznamu **Agent**.



Slika 37: Izbira agenta za simulacijo

- Za **Number of Episodes** uporabite **10**.
- Za **Max. Episode Length** uporabite **500**.
- Za simulacijo agenta kliknite **Simulate**.

Aplikacija odpre zavihek Simulation Session. Ko je simulacija končana, se v podoknu Simulation Results prikaže nagrada za vsako epizodo ter srednja vrednost nagrade in standardno odstopanje (Slika 38).



Slika 38: Rezultati simulacije učenja

V prikazanem primeru je agent za vse simulirane epizode dosegel največjo nagrado 500.

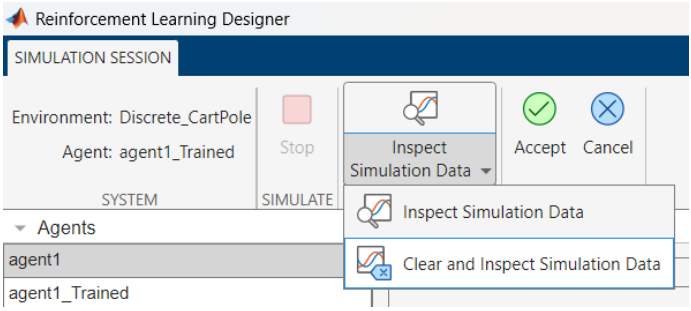
Opomba:

Če kateri izmed agentov največje nagrade ne more doseči, to nakazuje, da bi se lahko izboljšala odpornost naučenega agenta na različne začetne pogoje. To lahko dosežete z daljšim učenjem agenta, na primer z izbiro Average Window Length 10 namesto 5, kar prinese boljšo robustnost. Prav tako lahko spremenite nekatere možnosti agenta DQN, kot sta BatchSize in TargetUpdateFrequency, da spodbudite hitrejše in robustnejše učenje.

- Za analizo rezultatov simulacije kliknite **Inspect Simulation Data**.

Opomba:

Da iz podatkov Simulation Data Inspector počistite vse podatke, ki ste jih morda naložili v prejšnji seji v razdelku Inspect Simulation Data, izberite Clear and Inspect Simulation Data (Slika 39).

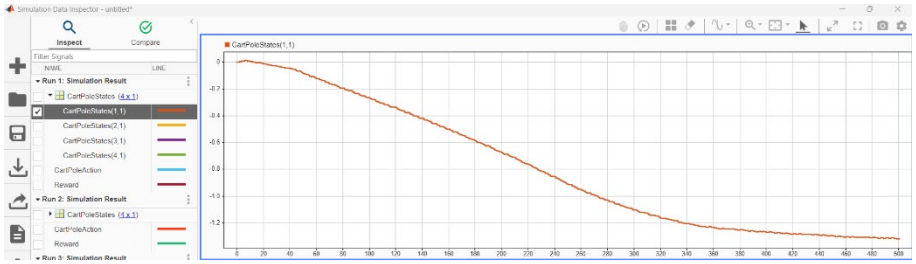


Slika 39: Dostop do analize rezultatov simulacije

V Simulation Data Inspector si lahko ogledate shranjene signale za vsako epizodo simulacije. Privzeto je izbrano eno območje izrisa.

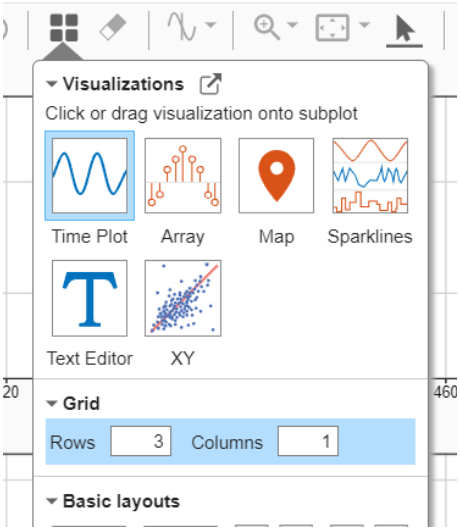
- Za prikaz položaja vozička, med prvo epizodo pod **Run 1: Simulation Result**, odprite spremenljivko **CartPoleStates** in izberite **CartPoleStates(1,1)**.

Voziček uspešno ostane v mejah vseh 500 sekund simulacije (Slika 40).



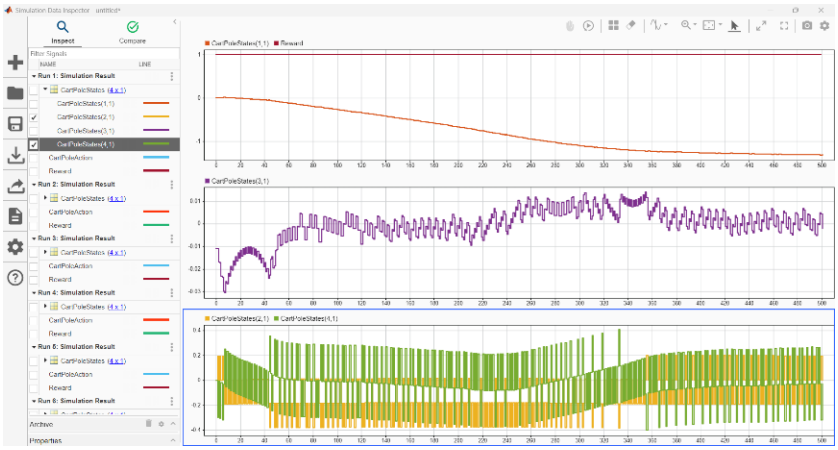
Slika 40: Prikaz rezultatov simulacije

- Za prikaz nagrade izberite spremenljivko **Reward**.
- Dodajte dve mesti za izris, tako da v ukazni vrstici kliknete **Vizualizations** in v odseku **Grid** za **Rows** vnesete **3** in za **Columns** vnesete **1** (Slika 41).



Slika 41: Razporeditev oken za primerjavo rezultatov simulacije

- V drugem izrisu prikažite kot nagiba **CartPoleStates(3,1)**.
- Izrišite tudi drugo in četrto stanje: hitrost vozička **CartPoleStates(2,1)** in kotno hitrost **CartPoleStates(4,1)**.



Slika 42: Pregled rezultatov simulacije

- Zaprite **Simulation Data Inspector**.
- Na zavihku **Simulation Session** kliknite **Accept,** da sprejmete rezultate simulacije.

V podoknu Results aplikacija doda strukturo rezultatov simulacije, experience1.

4.1.6 Izvozite agenta in shranite sejo

Izberite naučenega agenta in odprite ustrezen dokument agent1_Trained.

- Pod podoknom **Agents** dvokliknite **agent1_Trained**.
- Na zavihku **Reinforcement Learning** pod **Export** izberite **Agent**.
- Na zavihku **Reinforcement Learning** kliknite **Save Session**, da shranite sejo.

Če želite v prihodnosti nadaljevati delo, kjer ste končali, lahko sejo odprete v Reinforcement Learning Designerju.

4.1.7 Simulirajte agenta v ukazni vrstici

Za simulacijo agenta v ukazni vrstici najprej naložite okolje cart-pole.

➢ V **Command Window** vpišite in izvedite ukaz:

```
env = rlPredefinedEnv("CartPole-Discrete");
```

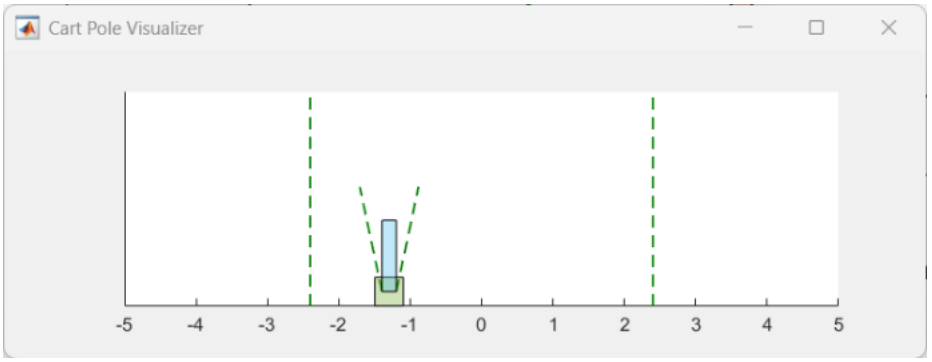
Okolje cart-pole ima vgrajen prikazovalnik okolja, ki vam omogoča, da vidite, kako se sistem obnaša med simulacijo in učenjem.

Izrišite okolje in izvedite simulacijo z uporabo naučenega agenta, ki ste ga predhodno izvozili iz aplikacije.

➢ V **Command Window** vpišite in izvedite ukaz:

```
plot(env)
xpr2 = sim(env,agent1_Trained);
```

Med simulacijo se v animaciji prikazuje gibanje vozička in palice.



Slika 43: Animacija balansiranja palice

Prikažite kumulativno nagrado za simulacijo.

➢ V **Command Window** vpišite in izvedite ukaz:

```
sum(xpr2.Reward)
```

4.2 Samostojno delo: Učenje z okrepitvijo

4.2.1 Robustnost agenta za Cart-Pole okolje

Cilj: Preverite, kako se naučeni agent DQN za Cart-Pole okolje odziva na različne začetne pogoje in ali zna ohranjati ravnovesje tudi pri daljših epizodah.

Navodila:

- Uporabite agenta, ki ste ga naučili v poglavju 4.1.
- V aplikaciji Reinforcement Learning Designer:
  - izberite agenta agent1_Trained,
- V zavihku Simulate nastavite:
  - Number of Episodes na 20,
  - Max Episode Length na 1000.
- Zaženite simulacijo in v zavihku Simulation Results opazujte, ali agent uspe ohraniti ravnotežje v vseh epizodah.
- Kliknite Inspect Simulation Data in preverite:
  - gibanje vozička,
  - kot palice,
  - trajanje epizod.

Opomba:

Sunkovit začetek ali rahlo nagnjena palica lahko povzročita padanje. S to nalogo ocenite robustnost naučenega agenta v različnih začetnih pogojih. V realnih aplikacijah je takšna robustnost nujna za zanesljivo delovanje sistemov.

4.2.2 Vpliv hiperparametrov in primerjava z algoritmom PPO

Cilj: Raziščite, kako sprememba hiperparametrov vpliva na delovanje agenta DQN in primerjajte rezultate z alternativnim algoritmom PPO (angl. Proximal Policy Optimization). Na ta način boste pridobili vpogled v pomen nastavitev učenja ter v primerjavo različnih pristopov učenja z okrepitvijo.

- Spremenite hiperparametre agenta, npr.:
  - BatchSize,
  - TargetUpdateFrequency,
  - AverageRewardWindowLength.
- Preverite, ali se zmogljivost izboljša.
- Ustvarite novega agenta z algoritmom PPO in primerjate rezultate.

4.2.3 Balansiranje žoge z robotom

Cilj: Preverite, ali se lahko agent, naučen s pomočjo algoritma SAC (Soft Actor-Critic), uspešno odzove na različne začetne položaje žoge na plošči. S tem boste raziskali sposobnost splošnega delovanja agenta v neznanih začetnih pogojih in zmožnost finega vodenja sistema z več vhodnimi dimenzijami.

Navodila:

- Odprite primer iz Matlab dokumentacije: *Train SAC Agent for Ball Balance Control*.
- V modelu za simulacijo spremenite začetne položaje žoge glede na naslednje koordinate:
  - X = 0,085 m in Y = 0,085 m od središča plošče,
  - X = 0,085 m in Y = 0,0 m od središča plošče in
  - X = 0,05 m in Y = 0,05 m od središča plošče.
- Simulirajte sistem za vsak položaj posebej.
- Opazujte:
  - ali žoga ostane v ravnovesju,
  - koliko časa traja stabilizacija,
  - ali pride do oscilacij ali padca žoge.

Opomba

Balansiranje žoge na plošči je zahtevna naloga vodenja z več prostostnimi stopnjami. Učenje z okrepitvijo omogoča agentu, da samodejno poišče optimalno strategijo nadzora. Z različnimi začetnimi pogoji preverite sposobnost posploševanja naučene politike.

Vir

MathWorks. (2024). Train SAC agent for ball balance control. Dostopno na: https://www.mathworks.com/help/reinforcement-learning/ug/train-sac-agent-for-ball-balance-control.html

Univerzitetna založba
Univerze v Mariboru

5 Prenosno učenje

Prenosno učenje (angl. Transfer Learning) je metoda strojnega učenja, ki ni strogo kategorizirana kot nadzorovano ali nenadzorovano učenje. To je tehnika, pri kateri se znanje modela, usposobljenega za eno nalogo, prenese na reševanje podobne naloge. Pogosto se uporablja v okviru globokega učenja, kjer se obstoječe nevronske mreže, trenirane na velikih zbirkah podatkov (npr. slikah, besedilu), prilagodijo novemu, običajno manjšemu, naboru podatkov.

Za razliko od klasičnih metod strojnega učenja, prenosno učenje omogoča, da model začne učenje z že pridobljenim znanjem, kar bistveno zmanjša količino potrebnih podatkov in časa za učenje. To omogoča učinkovitejše izvajanje tako nadzorovanih kot nenadzorovanih nalog.

V inženirstvu se prenosno učenje pogosto uporablja pri računalniškem vidu za vizualno kontrolo kakovosti, kjer se splošne mreže (npr. GoogLeNet, SqueezeNet, ResNet, VGG, EfficientNet) prilagajajo specifičnim nalogam (npr. prepoznavanje napak na izdelkih), pri obdelavi signalov z majhnimi količinami označenih podatkov (npr. vibracije, akustika), pri prenosu znanja med podobnimi stroji ali procesi v različnih obratih (npr. adaptacija modela na nov stroj), v robotiki, kjer se vedenjski vzorci ali zaznavanje, naučeno v simulaciji, prenesejo v realno okolje ipd.

Najpogosteje uporabljene metode vključujejo:

- Fino prilagajanje (angl. Fine Tuning): nadaljnje učenje že naučene mreže na novem sklopu podatkov.
- Zamrznitev slojev (angl. Feature Extraction): uporaba zgodnjih slojev kot fiksnih izluščevalcev značilk.
- Modeli s predhodnim učenjem (angl. Pretrained Models): uporaba že obstoječih arhitektur (npr. GoogLeNet, SqueezeNet, ResNet, MobileNet, BERT) kot osnove za nove naloge.

Prenosno učenje je še posebej uporabno, kadar imamo na voljo le omejeno količino označenih podatkov, kar je v inženirskih aplikacijah pogosto. Zaradi svoje učinkovitosti je postalo ključni pristop pri uporabi globokega učenja v praksi.

5.1 Prepoznava predmetov v realnem času

Primer prikazuje delovanje globoke konvolucijske nevronske mreže GoogLeNet za klasifikacijo predmetov v realnem času. GoogLeNet je bil naučen na več kot milijon slikah in lahko razvrsti predmete na slikah v 1000 kategorij (kot so tipkovnica, skodelica za kavo, svinčnik in številne živali). Mreža vzame sliko kot vhod in vrne oznako za predmet na sliki skupaj z verjetnostjo za pripadnost kategoriji.

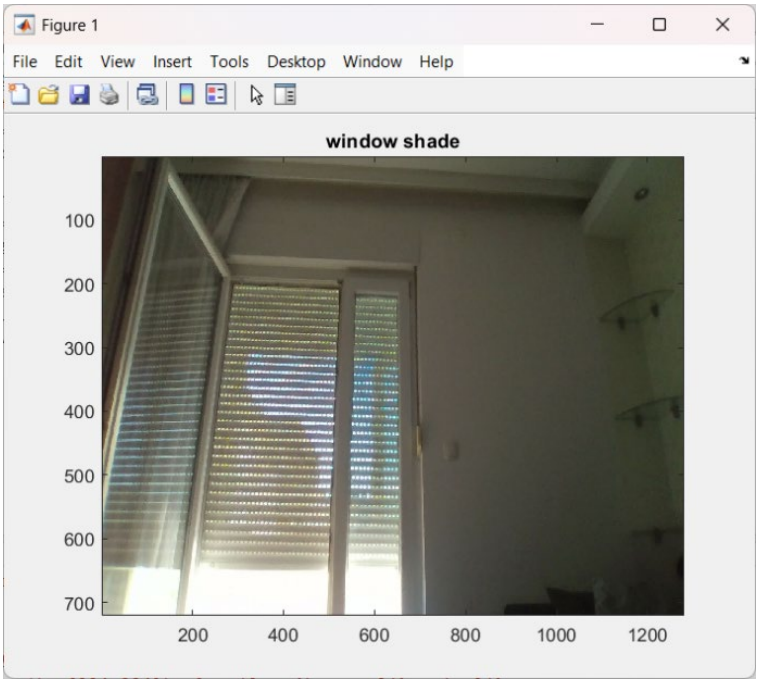
- Ustvarite nov skript z imenom *strojni_vid_GoogLeNet.m*.
- V odprt skript prilepite spodnjo kodo.
- Kodo zaženite s klikom na gumb **Run** ali s pritiskom tipke **F5**.

```
camera = webcam;                    % povezava kamere
net = googlenet;                    % nalaganje nevronske mreže

while true
    im = snapshot(camera);        % zajem slike
    image(im);                    % prikaz slike
    im = imresize(im,[224 224]); % prilagoditev velikosti slike za
squeezenet
    label = classify(net,im);     % razvrstitev slike
    title(char(label));           % prikaz kategorije slike
    drawnow
end
```

Skript v zanki v realnem času zajema slike s kamere, jih obdeluje in klasificira z vnaprej naučeno nevronsko mrežo GoogLeNet. Najprej se vzpostavi povezava s kamero (*webcam*) in naloži nevronska mreža (*googlenet*). Znotraj neskončne zanke se vsakič zajame slika (*snapshot*), prikaže na zaslonu (*image*), spremeni njena velikost na 224 × 224 slikovnih pik (kar zahteva GoogLeNet), ter nato klasificira s pomočjo funkcije *classify*. Rezultat klasifikacije se prikaže kot naslov slike (*title*). Funkcija *drawnow* omogoča sproten prikaz rezultatov.

Slika 44 prikazuje rezultat kode, ki zajema sliko s kamere in jo v realnem času klasificira z GoogLeNet. Nevronska mreža je prizor prepoznala kot window shade (okensko senčilo), kar se izpiše kot naslov nad prikazano sliko. Sistem deluje tako, da vsako zajeto sliko zmanjša na 224 × 224 slikovnih pik, jo razvrsti in rezultat sproti prikaže.



Slika 44: Prepoznava predmetov na slikah v realnem času z mrežo GoogLeNet

- Za prekinitev neskončne zanke pritisnite Ctrl + C.

5.2 Prepoznava predmetov na sliki

Ta primer prikazuje, kako razvrstiti sliko z uporabo vnaprej usposobljene globoke konvolucijske nevronske mreže SqueezeNet.

5.2.1 Ustvarite nov skript s prednaučeno mrežo SqueezeNet

- Ustvarite nov skript z imenom *strojni_vid_SqueezeNet.m*.
- V odprt skript prilepite spodnjo kodo.

```
% Naloži prednaučeno mrežo SqueezeNet
net = squeezenet;
```

Ukaz naloži naučen model SqueezeNet, ki je majhna in hitra konvolucijska nevronska mreža za klasifikacijo slik. Pogosto se uporablja kot alternativa GoogLeNet z manj parametri in hitrejšim delovanjem ter je primerna za uporabo v realnem času ali na napravah z omejenimi viri.

5.2.2 Naložite sliko in spremenite velikost slike

Slika, ki jo želite razvrstiti, mora biti enake velikosti, kot je predvidena vhodna velikost mreže. Za SqueezeNet je velikost vnosa v mrežo določena z lastnostjo InputSize.

Slika 45: Slika Test00.jpg za učenje mreže SqueezeNet

Preberite sliko, ki jo želite razvrstiti in ji spremenite velikost. Spreminjanje velikosti nekoliko spremeni razmerje stranic slike. Uporabite lahko sliko mavričnega dežnika (Slika 45).

➢ Dodajte naslednje vrstice.

```
% Preberi sliko in prilagodi velikost
I = imread("test00.jpeg");
inputSize = net.Layers(1).InputSize;
I = imresize(I,inputSize(1:2));
```

Koda najprej naloži sliko iz datoteke z ukazom *imread*, nato s pomočjo *net.Layers(1).InputSize* prebere zahtevano velikost vhodnega sloja izbrane nevronske mreže (npr. 227 × 227 za SqueezeNet), in na koncu z *imresize* ustrezno prilagodi dimenzije slike (širina in višina), da jo je možno uporabiti kot vhod v mrežo za klasifikacijo.

5.2.3 Razvrstite in prikažite sliko

Razvrstite in prikažite sliko s predvideno oznako.

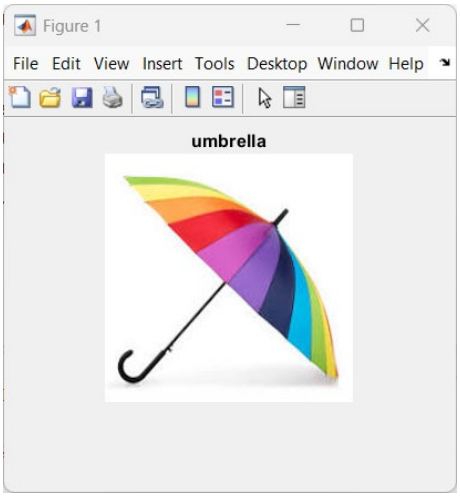
➢ Dodajte naslednje vrstice.

```
% Prikaži sliko in izpiši pripadnost kategoriji
label = classify(net,I);
figure
imshow(I)
title(string(label))
```

Koda najprej s pomočjo *classify(net, I)* izvede klasifikacijo slike *I* z nevronsko mrežo *net* in shrani napovedano oznako v spremenljivko *label*. Nato s *figure* odpre novo grafično okno, z *imshow(I)* prikaže sliko, in s *title(string(label))* doda nad sliko besedilo z rezultatom razvrstitve (npr. umbrella). Tako omogoča vizualni prikaz slike skupaj s prepoznano kategorijo.

➢ Kodo zaženite s klikom na gumb **Run** ali s pritiskom tipke **F5**.

Algoritem pravilno kategorizira predmet na sliki kot dežnik (Slika 46).



Slika 46: Pravilno kategoriziran dežnik

5.2.4 Naložite in kategorizirajte novo sliko

Spremenite programsko kodo tako, da naloženo sliko dežnika *test00.jpeg* zamenjate s sliko industrijskega robota *test01.jpeg*.

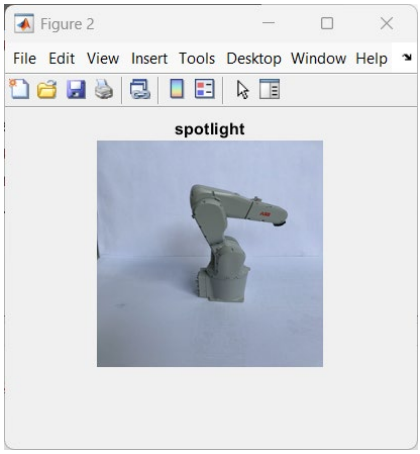
Slika 47: Slika za učenje Test01.jpg

➢ Spremenite vrstico za nalaganje slike.

```
I = imread("test01.jpeg");
```

➢ Kodo zaženite s klikom na gumb **Run** ali s pritiskom tipke **F5**.

Upoštevajte, da algoritem predmeta ne prepozna kot industrijskega robota (Slika 48).



Slika 48: Nepravilno kategoriziran predmet industrijski robot

5.3 Prilagodite mrežo za prepoznavo novih predmetov

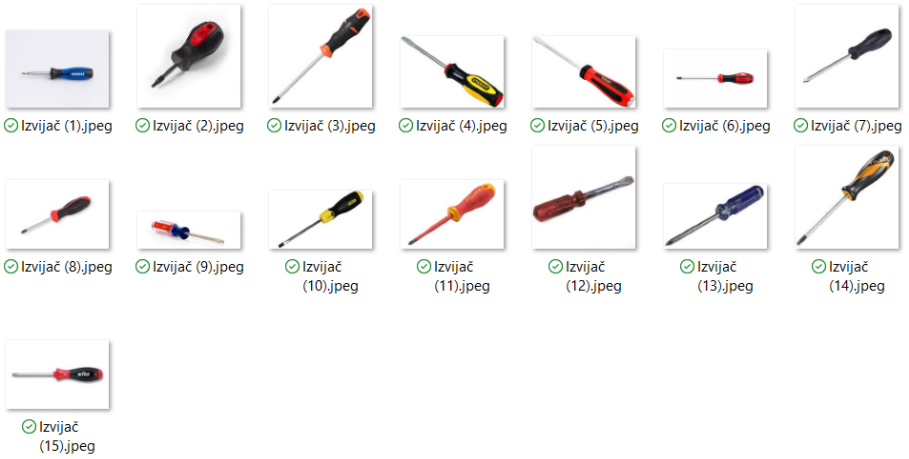
V tem primeru boste vnaprej naučeno mrežo prilagodili za prepoznavo novih predmetov.

Naloga je za skripte prirejena po viru: *Retrain Neural Network to Classify New Images (www.mathworks.com)*.

5.3.1 Naložite slike

Za to nalogo naložite v MATLAB lastne slike predmetov, ki jih želite analizirati. To so lahko industrijski stroji, orodja, izdelki ali drugi objekti, ki jih obstoječe nevronske mreže (npr. SqueezeNet) še ne prepoznajo.

Za izbrani razred predmetov pripravite več slik, posnetih iz različnih zornih kotov in pri različnih svetlobnih pogojih (Slika 49). S tem boste zagotovili, da bo predmet zajet čim bolj splošno, kar je ključno za nadaljnjo uporabo v strojnem vidu, razvrščanju ali učenju novih modelov. Vse slike poimenujte z nazivom kategorije predmeta na sliki, saj bo v algoritem v fazi učenja iz imena datoteke pridobil razred predmeta.

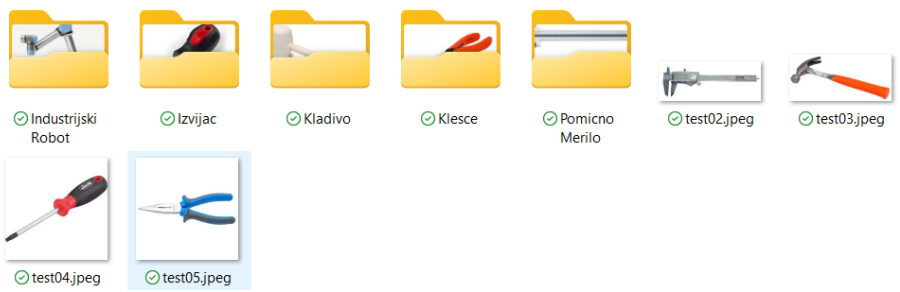


Slika 49: Kategorizirane slike izvijačev za prenosno učenje

Namig:

Slike organizirajte v poimenovane mape, pri čemer naj ime vsake mape ustreza kategoriji oziroma predmetu, ki ga prikazuje (npr. industrijski robot, izvijač, kladivo). Takšna struktura vam bo omogočila lažji nadzor nad podatki in učinkovitejše delo pri pripravi učne množice. Ker za kakovostno prepoznavo potrebujete izjemno veliko količino slik, vam bo dobra organizacija in dosledno poimenovanje datotek močno olajšalo nadaljnjo obdelavo (Slika 50).

Pomembno je, da nekaj slik iz vsake kategorije izločite za kasnejše testiranje naučenega modela. Te slike ne smejo biti uporabljene v procesu prenosnega učenja, saj se bo model z njimi soočil šele v fazi testiranja. Na ta način boste lahko objektivno preverili, kako dobro se model odziva na nove, še nevidene primere, in ocenili njegovo splošnost in robustnost.



Slika 50: Nove slike za prenosno učenje, organizirane v mapah.

Tako organizirane slike (lahko v formatu ZIP) prenesite v delovno mapo projekta.

- Povlecite in spustite datoteko *slike.zip* v polje **Files** in počakajte, da se podatki prenesejo.
- Kliknite **OK**.
- V **Comand Window** vnesite in izvedite ukaz:

```
unzip("slike.zip");
```

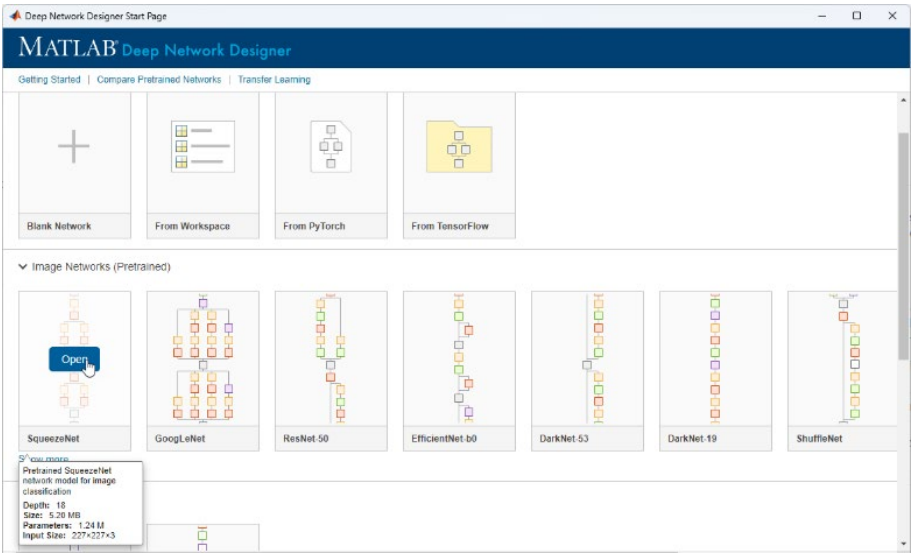
5.3.2 Odprite aplikacijo Deep Network Designer in izberite SqueezeNet

- V **Comand Window** vnesite in izvedite ukaz:

```
deepNetworkDesigner("-v1")
```

Ukaz odpre prvo različico orodja Deep Network Designer (Slika 51), ki omogoča grafično gradnjo, urejanje in analiziranje nevronskih mrež.

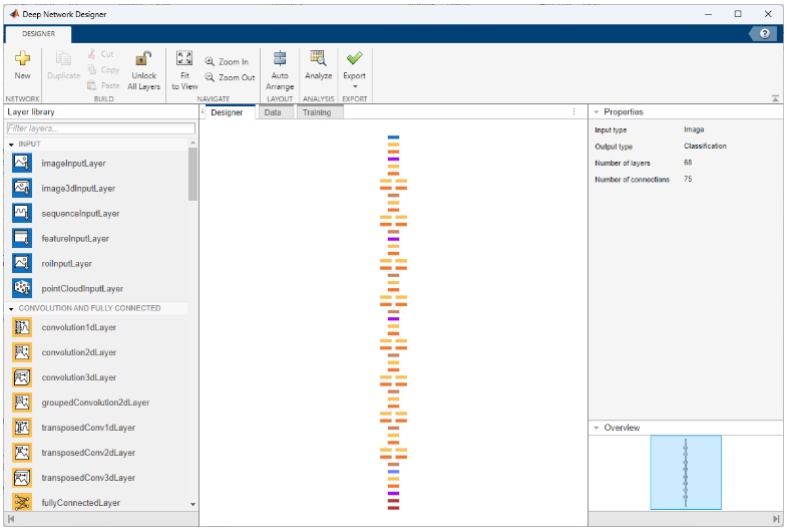
- Na seznamu vnaprej naučenih mrež izberite **SqueezeNet** in kliknite **Open**.



Slika 51: Nabor vnaprej naučenih nevronskih mrež

5.3.3 Raziščite mrežo

Deep Network Designer prikaže pomanjšan pogled na celotno mrežo v podoknu Designer (Slika 52).



Slika 52: Strukturna globoke nevronske mreže SqueezeNet

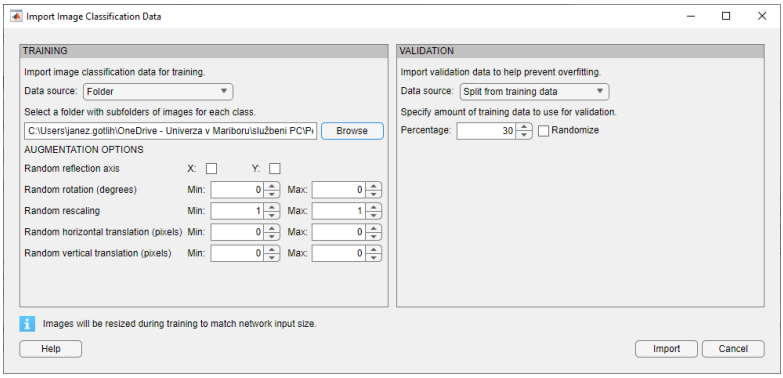
Raziščite strukturo in plasti mreže. Za povečavo z miško uporabite tipko Ctrl in kolesce na miški. Za premikanje uporabite smerne tipke ali držite kolesce za pomikanje in povlecite miško.

- Izberite plast, da si ogledate njene lastnosti.
- Prekličite izbor vseh plasti, da si ogledate povzetek mreže v podoknu **Properties**.

5.3.4 Uvozite podatke

Naložite podatke v Deep Network Designer (Slika 53).

- Na zavihku **Data** kliknite **Import Data** in **Import Image Classification Data**.
- Na seznamu **Data source** izberite **Folder**.
- Kliknite **Browse** in izberite mapo, ki vsebuje vaše slike.



Slika 53: Uvoz novih slik v aplikacijo za prenosno učenje

5.3.5 Povečajte količino učnih podatkov s transformacijami slik

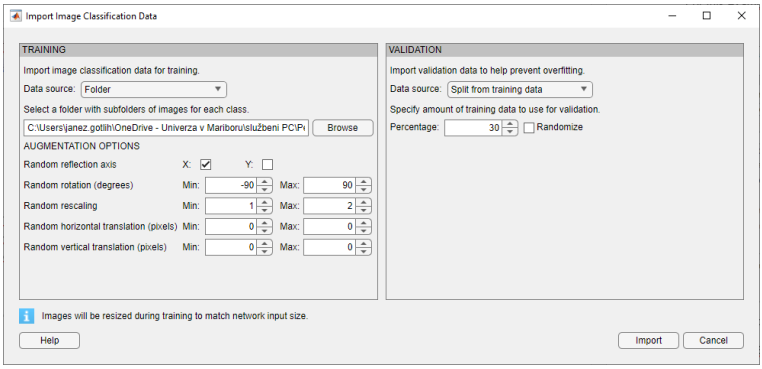
Za povečanje učne množice za učenje lahko uporabite spremenjene slike. Aplikacija Deep Network Designer nudi naslednje možnosti:

- naključno zrcaljene preko osi *X*,
- naključno zrcaljene preko osi *Y*,

- naključna rotacija,
- naključno spreminjanje velikosti,
- naključna vodoravna translacija,
- naključna navpična translacija.

Količino učnih podatkov lahko učinkovito povečate tako, da za svoje slike uporabite naključno povečanje. Razširitev vam omogoča tudi, da naučite mrežo, da je neobčutljiva na popačenja v slikovnih podatkih. Vhodnim slikam lahko na primer dodate naključne rotacije, tako da je mreža neobčutljiva na prisotnost rotacije v vhodnih slikah.

➢ Za ta primer uporabite:
- naključno zrcaljenje preko osi **X**,
- naključno rotacijo iz obsega **[-90,90]** in
- naključno spreminjanje velikosti v merilu **[1,2]** kot prikazuje Slika 54.

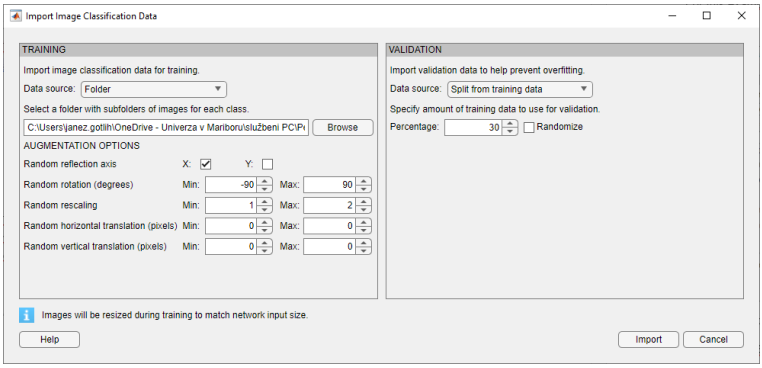


Slika 54: Transformacije slike za povečanje količine učnih podatkov

5.3.6 Validacijski podatki

Podatke za preverjanje lahko izberete tako, da jih ločite od podatkov za učenje, ali pa jih uvozite iz drugega vira (Slika 55). Validacija oceni zmogljivost modela na novih podatkih v primerjavi s podatki za učenje ter omogoči spremljanje učinkovitosti učenja in zaščititi pred čezmernim ujemanjem.

➢ V tem primeru za preverjanje uporabite **30** odstotkov slik.

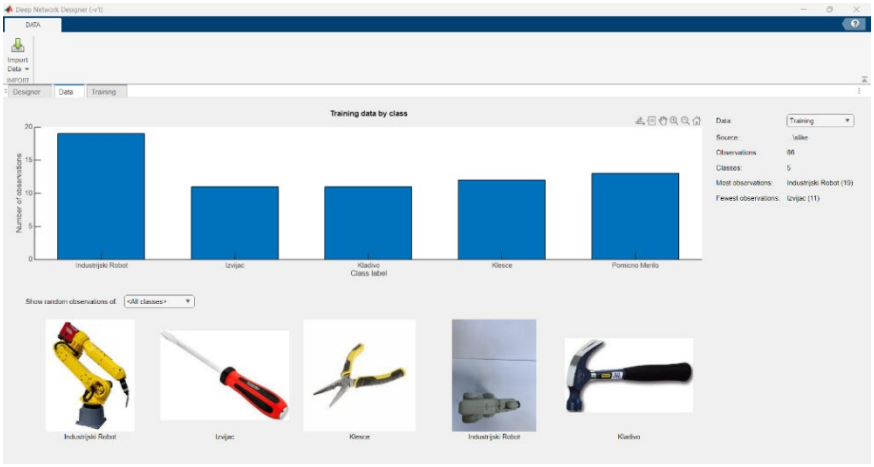


Slika 55: Razdelitev podatkov na učni in validacijski niz

➢ Kliknite **Import**, da uvozite podatke v **Deep Network Designer**.

5.3.7 Preglejte podatke

Z uporabo aplikacije Deep Network Designer lahko vizualno pregledate porazdelitev podatkov za učenje in validacijo na zavihku Data. Vidite lahko, da je v tem primeru v naboru podatkov pet razredov. Ogledate si lahko tudi naključne vzorce iz vsakega razreda (Slika 56).



Slika 56: Predogled podatkov za prenosno učenje

5.3.8 Pripravite mrežo za učenje

Uredite mrežo v podoknu Designer, da določite novo število razredov v svojih podatkih. Če želite mrežo pripraviti za prenosno učenje, uredite zadnjo učljivo plast in končno klasifikacijsko plast. Če želite uporabiti vnaprej naučeno mrežo za prenosno učenje, morate spremeniti število razredov, da se bo ujemalo z vašim novim naborom podatkov.

- Poiščite zadnjo učljivo plast v mreži.

Za SqueezeNet je zadnja učljiva plast zadnja konvolucijska plast, 'conv10'.

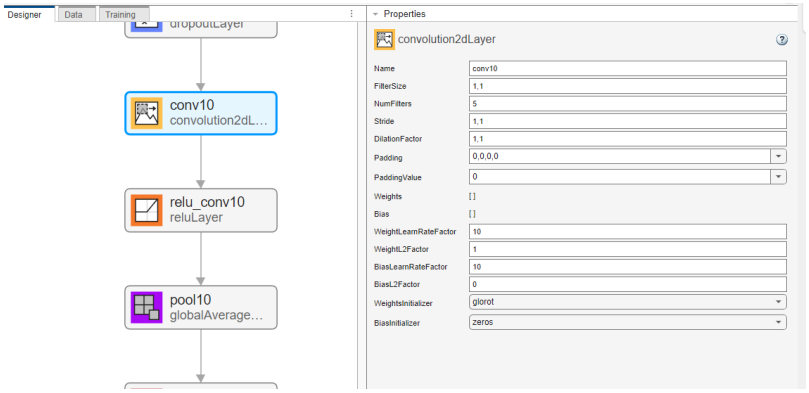
- Izberite plast **'conv10'**.
- Na dnu podokna **Properties** kliknite **Unlock Layer**.
- V ukaznem oknu z opozorilom, ki se prikaže, kliknite **Unlock Anyway**.

S tem odklenete lastnosti plasti, da jih lahko prilagodite svoji novi nalogi.

- V novi konvolucijski 2D-plasti nastavite **FilterSize na [1 1]**.

Lastnost NumFilters določa število razredov za klasifikacijo.

- Spremenite **NumFilters** na število razredov v novih podatkih, v tem primeru **5**.
- Spremenite stopnjo učenja tako, da bo učenje v novi plasti hitrejše kot v prenesenih slojih, tako da nastavite **WeightLearnRateFactor** in **BiasLearnRateFactor** na **10**.

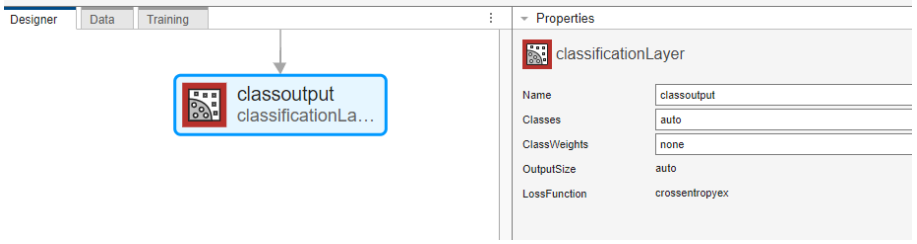


Slika 57: Nastavitve parametrov učne plasti za prenosno učenje

Uredite izhodni sloj

- Izberite končno klasifikacijsko plast, **ClassificationLayer_predictions**.
- Kliknite **Unlock Layer** in nato **Unlock Anyway**.

Za odklenjeno izhodno plast vam ni treba nastaviti OutputSize. V času učenja Deep Network Designer samodejno nastavi izhodne razrede plasti iz podatkov (Slika 58).

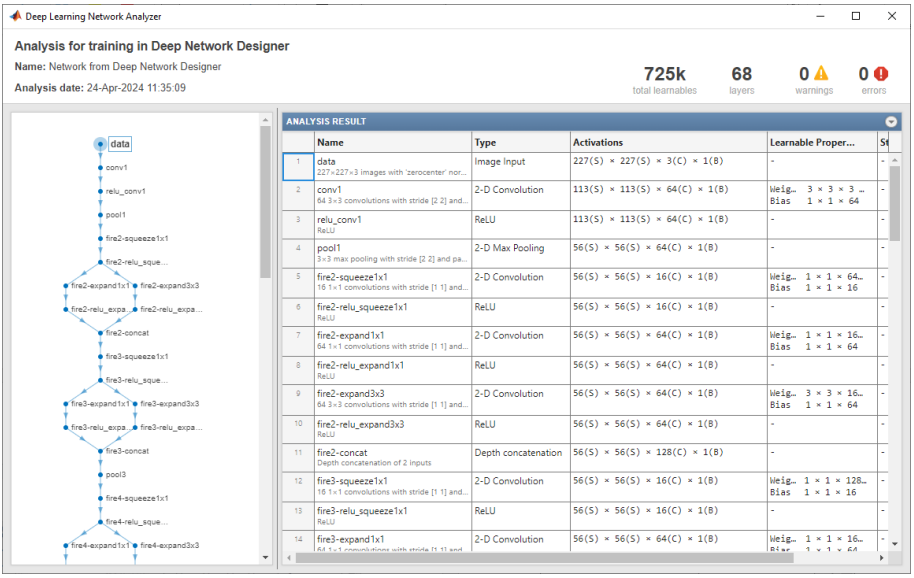


Slika 58: Nastavitve parametrov izhodne plasti za prenosno učenje

5.3.9 Preverite mrežo

- Da preverite, ali je mreža pripravljena za učenje, kliknite **Analyze**.

Če Deep Learning Network Analyzer ne sporoči napak, je urejena mreža pripravljena za učenje (Slika 59).



Slika 59: Mreža, pripravljena za učenje

5.3.10 Naučite mrežo

V Deep Network Designerju lahko naučite mrežo, uvoženo ali ustvarjeno v aplikaciji.

➢ Za učenje mreže s privzetimi nastavitvami na zavihku **Training** kliknite **Train**.

Privzete možnosti učenja so bolj primerne za velike nabore podatkov kot za majhne, saj uporabljajo višjo začetno učno hitrost in večje število epoh, kar pri majhnih naborih podatkov pogosto vodi v prenaučenost (angl. overfitting). Za večji nadzor nad učenjem v primeru majhnega nabora podatkov prilagodite nastavitve za učenje.

- Nastavite Initial Learn Rate na majhno vrednost, da upočasnite učenje v prenesenih slojih.
- ValidationFrequency nastavite tako, da se validacija izvede enkrat na epoho (ali pogosteje pri majhnih naborih podatkov).
- Določite majhno število epoch. Epoha ali obdobje je celoten cikel učenja na celotnem nizu podatkov za učenje. Za prenosno učenje vam ni treba učiti toliko obdobij. V primeru heterogenih podatkov je lahko potrebnih več epoh učenja,

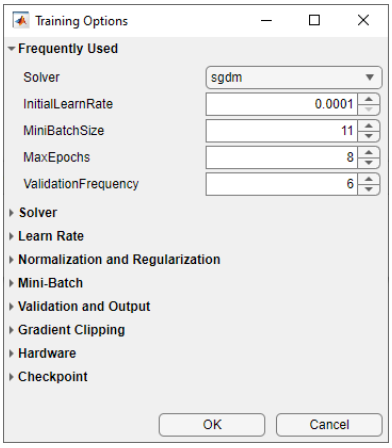
pri čemer se za preprečevanje prenaučenosti uporablja zgodnje ustavljanje (parameter ValidationPatience).

– Določite MiniBatchSize, to je, koliko slik želite uporabiti v vsaki ponovitvi. Če želite zagotoviti, da se v vsakem obdobju uporablja celoten nabor podatkov, nastavite MiniBatchSize tako, da enakomerno razdelite število učnih vzorcev.

Za primer z majhnim naborom slik prilagodite Training Options.

➢ Kliknite **Training Options** in nastavite naslednje vrednosti:
  – **InitialLearnRate** na **0,0001**,
  – **MaxEpochs** na **8**,
  – **ValidationFrequency** na **6**,
  – **MiniBatchSize** na **11**.

Nastavitve smo izbrali za primer 66 učnih vzorcev, z namenom da enakomerno razdelimo podatke za učenje in zagotovimo, da bomo v vsaki epohi uporabili celoten nabor podatkov za učenje (Slika 60).

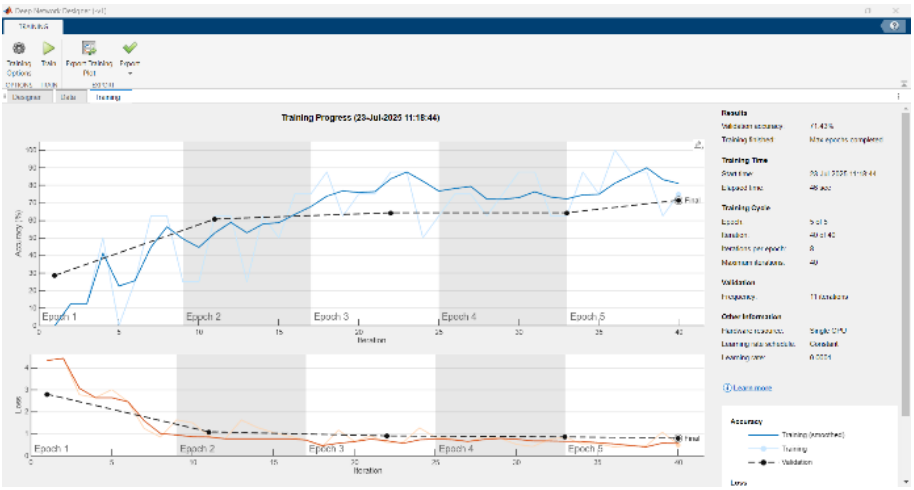


Slika 60: Nastavitve možnosti učenja

Nastavitev parametra MiniBatchSize 11, pomeni, da se bo mreža vsak korak učila na 11 slikah, za eno epoho bo potrebovala šest iteracij učenja (66/11 = 6). ValidationFrequency 6, pomeni, da se validacija izvede po vsakih šestih iteracijah, torej enkrat na vsako epoho.

➢ Za učenje mreže z izbranimi možnostmi kliknite **OK** in nato kliknite **Train**.

Deep Network Designer omogoča vizualizacijo in spremljanje napredka pri učenju (Slika 61). Nato lahko uredite možnosti učenja in po potrebi ponovno učite mrežo.



Slika 61: Prikaz napredka učenja pri prenosnem učenju

5.3.11 Izvozite rezultate in ustvarite programsko kodo

Izvozite arhitekturo mreže s treniranimi utežmi.

➢ Na zavihku **Training** izberite **Export** in **Export Trained Network and Results**.

Deep Network Designer izvozi naučeno mrežo kot spremenljivko trainedNetwork_1 in informacije o učenju kot spremenljivko trainInfoStruct_1.

➢ V **Comand Window** vnesite in izvedite ukaz:

```
trainInfoStruct_1
```

Ustvarite lahko tudi Matlab kodo, ki poustvari mrežo in uporabljene možnosti učenja.

➢ Na zavihku **Training** izberite **Export** in **Generate Code for Training**.

Preglejte kodo, da se naučite, kako programsko pripraviti podatke za učenje, ustvariti arhitekturo mreže in naučiti mrežo.

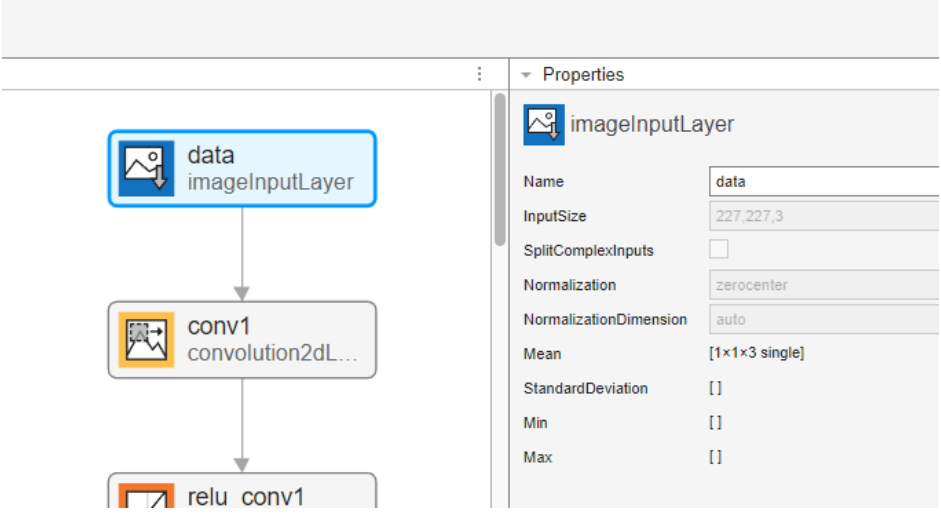
5.3.12 Razvrstite novo sliko

Naložite novo sliko za razvrščanje z uporabo naučene mreže.

➢ V **Comand Window** vnesite in izvedite ukaz:

```
I = imread("test01.jpeg");
```

Deep Network Designer med učenjem spremeni velikost slik, da se ujema z zahtevano velikostjo vnosa v mrežo. Če si želite ogledati velikost vnosa v mrežo, pojdite v podokno Designer in izberite imageInputLayer (prvi sloj). Mreža SqueezeNet ima vhodno velikost 227 x 227 (Slika 62).



Slika 62: Prikaz lastnosti vhodne plasti v mrežo SqueezNet

Spremenite velikost testne slike, da bo ustrezala velikosti, primerni za vnos v mrežo.

- V **Comand Window** vnesite in izvedite ukaz:

```
I = imresize(I, [227 227]);
```

Razvrstite testno sliko z uporabo naučene mreže.

- V **Comand Window** vnesite in izvedite ukaze:

```
[YPred,probs] = classify(trainedNetwork_1,I);
imshow(I)
label = YPred;
title(string(label) + ", " + num2str(100*max(probs),3) + "%");
```

Mreža pravilno kategorizira novo sliko kot industrijski robot (Slika 63).

Slika 63: Pravilna kategorizacije slike industrijskega robota

5.4 Samostojno delo: Prenosno učenje

5.4.1 Razvrstitev testnih slik z uporabo naučenega modela

Cilj: Preverite natančnost že naučenega modela na vseh testnih slikah, ki niso bile uporabljene v procesu učenja.

Navodila:

- Izvedite razvrščanje preostalih testnih slik, ki so bile med učenjem izločene.
- Primerjajte napovedano oznako z dejansko.
- Zabeležite število pravilno in nepravilno razvrščenih slik.
- Izdelajte matriko zmede.

5.4.2 Priprava lastnega nabora slik za novo kategorijo

Cilj: Dodajte povsem novo kategorijo predmetov (npr. industrijski del, orodje, komponenta) in prilagodite obstoječo mrežo, da jih prepozna.

Navodila:

- Zberite vsaj 20 slik za novo kategorijo (lahko jih fotografirate ali uporabite slike s spleta).
- Organizirajte slike v ustrezno mapo.
- Uporabite obstoječo arhitekturo (npr. SqueezeNet) in ponovno naučite mrežo z dodatno kategorijo.
- Testirajte, ali mreža novo kategorijo prepozna pravilno.

5.4.3 Primerjava različnih arhitektur (SqueezeNet, GoogLeNet)

Cilj: Preverite, katera arhitektura bolje razvršča vaše podatke (natančnost, hitrost, velikost modela).

Navodila:

- Naučite isti nabor slik z dvema različnima arhitekturama (npr. SqueezeNet in GoogLeNet).

- Primerjajte:
  - natančnost modela,
  - trajanje učenja,
  - velikost modela,
  - robustnost na testnih podatkih.

Opomba:

Vsaka arhitektura ima različne prednosti, SqueezeNet je manjša in hitrejša, GoogLeNet je natančnejša, a večja.

5.4.4 Uporaba funkcije classify iz ukazne vrstice

Cilj: Preverite, kako uporabiti naučeno mrežo programsko, brez grafičnega uporabniškega vmesnika (angl. GUI), in jo vključiti v avtomatski sistem.

Navodila:

- Naložite sliko iz ukazne vrstice (imread).
- Prilagodite velikost slike (imresize).
- Uporabite classify(trainedNetwork, I) za napoved.
- Prikaz rezultatov dopolnite z imshow in title.

Opomba:

Tak pristop je prvi korak k avtomatizaciji, saj omogoča uporabo mreže v realnih aplikacijah brez uporabniškega vmesnika.

6 Sklep

Skripta celovito predstavi področje strojnega učenja v kontekstu inženirstva in povezujejo temeljne koncepte z uporabo v realnih primerih. Bralec postopoma spoznava različne pristope strojnega učenja, kot so nadzorovano učenje, nenadzorovano učenje, učenje z okrepitvijo in prenosno učenje, ter razume njihove posebnosti, prednosti in omejitve. Vsaka metoda je predstavljena s pomočjo praktičnih nalog in primerov, ki temeljijo na resničnih inženirskih problemih, kot so klasifikacija vibracijskih signalov, napoved obrabe orodij, uravnoteženje sistemov ter prepoznavanje predmetov na slikah.

Poudarek je na praktični uporabi orodij in aplikacij v okolju MATLAB, kot so Classification Learner, Regression Learner, Reinforcement Learning Designer in Deep Network Designer. Uporaba teh orodij omogoča študentom, da teoretične koncepte preizkusijo na podatkih, razvijejo lastne modele in jih ocenijo z ustreznimi metrikami brez poglobljenega znanja programiranja. Posebno vrednost predstavlja vključitev eksperimentalnih podatkov ter nalog, ki spodbujajo kritično razmišljanje, raziskovanje in samostojno delo.

Skripta ne služijo le kot učni pripomoček za razumevanje strojnega učenja, temveč tudi kot izhodišče za razvoj uporabnih rešitev v industriji. Študentom tehniških smeri omogočajo, da pridobijo kompetence za obvladovanje podatkovno podprtih pristopov, ki postajajo nepogrešljiv del sodobnega inženirskega dela. Ob tem se

razvija tudi razumevanje pomena kakovostne priprave podatkov, izbire ustreznih modelov ter interpretacije rezultatov v luči tehničnih in praktičnih zahtev.

Povezava med teoretičnim znanjem in praktično uporabo, ki jo ta skripta vzpostavljajo, prispeva k razvoju inženirskega razmišljanja ter pripravljenosti na reševanje kompleksnih nalog v industrijskem okolju. Tak pristop krepi tudi zavedanje o pomenu umetne inteligence kot orodja za podporo odločanju, optimizacijo in avtomatizacijo procesov v sodobni proizvodnji, logistiki, vzdrževanju in drugih inženirskih panogah.

Vabim te, da pridobljeno znanje s področja strojnega učenja uporabiš tudi pri drugih predmetih, projektnih nalogah ali raziskovalnem delu. Metode, ki si jih spoznal v teh skriptih, se pojavljajo v najrazličnejših kontekstih, od tehničnih in proizvodnih sistemov do ekonomskih, okoljskih ali organizacijskih odločitev. Povsod tam, kjer si prizadevamo za boljše, hitrejše, učinkovitejše ali zanesljivejše rešitve, ima strojno učenje pomembno vlogo. Ključno pa je, da znaš prepoznati takšne situacije in znanje, ki si ga pridobil, ustrezno prenesti v prakso.

Univerzitetna založba
Univerze v Mariboru

Literatura

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. Springer.
Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. MIT press.
Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer.
Kuhn, M., & Johnson, K. (2016). *Applied predictive modeling*. Springer.
MathWorks. Statistics and Machine Learning Toolbox Documentation. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/stats/
MathWorks. Deep Learning Toolbox Documentation. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/deeplearning/
MathWorks. Reinforcement Learning Toolbox Documentation. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/reinforcement-learning/
MathWorks. Transfer Learning with Deep Network Designer. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/deeplearning/ug/transfer-learning.html
MathWorks. Build Condition Model for Industrial Machinery and Manufacturing Processes. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/stats/build-condition-model-for-industrial-machinery-and-manufacturing-processes.html
MathWorks. Three-Axis Vibration Data Set for Anomaly Detection. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/predmaint/anomaly-detection.html
MathWorks. (2024). Design and Train Agent Using Reinforcement Learning Designer. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/reinforcement-learning/ug/design-dqn-using-rl-designer.html
MathWorks. Train DQN Agent to Balance Cart-Pole System. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/reinforcement-learning/ug/train-dqn-agent-to-balance-cart-pole-system.html
MathWorks. Retrain Neural Network to Classify New Images. Dostopno 1. 10. 2025 na: https://www.mathworks.com/help/deeplearning/ug/retrain-neural-network-to-classify-new-images.html
Wikipedia. Confusion matrix. Dostopno 1. 10. 2025 na: https://en.wikipedia.org/wiki/Confusion_matrix
Wikipedia. Principal component analysis. Dostopno 1. 10. 2025 na: https://en.wikipedia.org/wiki/Principal_component_analysis

Priloge

Priloga 1

Priloga vsebuje eksperimentalne podatke o varjenju, ki vključujejo tok, podajalno hitrost in izmerjeno površino vara. Namenjeni so analizi povezav med procesnimi parametri in lastnostmi zvara.

Tok [A]	Podajalna hitrost [cm/min.]	Površina vara [mm^2]
65	20	5,204
75	20	5,688
85	20	5,615
95	20	12,153
105	20	12,434
115	20	11,177
125	20	9,963
135	20	10,773
145	20	19,924
155	20	24,063
65	25	4,895
75	25	6,951
85	25	7,126
95	25	8,509
105	25	10,249
115	25	12,321
125	25	11,752
135	25	9,613
145	25	14,407
155	25	17,807
65	30	3,359
75	30	4,240
85	30	5,537
95	30	5,210
105	30	7,851
115	30	6,103
125	30	6,618
135	30	8,167
145	30	11,676
155	30	16,176
65	35	2,097
75	35	2,361
85	35	3,380
95	35	9,285
105	35	5,573
115	35	4,380
125	35	4,586

Tok [A]	Podajalna hitrost [cm/min.]	Površina vara [mm^2]
135	35	6,608
145	35	9,285
155	35	9,405
65	40	2,041
75	40	2,017
85	40	2,493
95	40	3,193
105	40	4,594
115	40	7,005
125	40	7,195
135	40	8,216
145	40	10,182
155	40	11,308

Priloga 2

Priloga vsebuje podatke o obrabi orodja pri struženju, namenjene analizi z metodami gručenja. Vključujejo tako številske značilke (npr. rezalna hitrost, podajalna hitrost, globina rezanja, čas obdelave, trdota) kot tudi kategorične oznake obrabe. Ta kombinacija omogoča uporabo naprednih metod za raziskovanje podatkovne strukture in prepoznavanje skupin z namenom boljšega razumevanja vpliva vhodnih parametrov na obrabo orodja.

Rezalna hitrost [m/min.]	Podajalna hitrost [mm/vrt]	Globina rezanja [mm]	Čas obdelave [s]	Trdota [HRC]	Obraba orodja
100	0,08	0,25	2	56	zmerna
150	0,08	0,25	2	56	zmerna
220	0,08	0,25	2	56	zmerna
100	0,12	0,25	2	56	zmerna
150	0,12	0,25	2	56	zmerna
220	0,12	0,25	2	56	zmerna
100	0,16	0,25	2	56	zmerna
150	0,16	0,25	2	56	zmerna
220	0,16	0,25	2	56	zmerna
100	0,08	0,5	2	56	zmerna
150	0,08	0,5	2	56	zmerna
220	0,08	0,5	2	56	zmerna
100	0,12	0,5	2	56	zmerna
150	0,12	0,5	2	56	zmerna
220	0,12	0,5	2	56	zmerna
100	0,16	0,5	2	56	zmerna
150	0,16	0,5	2	56	zmerna
220	0,16	0,5	2	56	zmerna
100	0,08	0,25	4	56	zmerna
150	0,08	0,25	4	56	zmerna
220	0,08	0,25	4	56	zmerna

Rezalna hitrost [m/min.]	Podajalna hitrost [mm/vrt]	Globina rezanja [mm]	Čas obdelave [s]	Trdota [HRC]	Obraba orodja
100	0,12	0,25	4	56	zmerna
150	0,12	0,25	4	56	zmerna
220	0,12	0,25	4	56	zmerna
100	0,16	0,25	4	56	zmerna
150	0,16	0,25	4	56	zmerna
220	0,16	0,25	4	56	zmerna
100	0,08	0,5	4	56	zmerna
150	0,08	0,5	4	56	zmerna
220	0,08	0,5	4	56	zmerna
100	0,12	0,5	4	56	zmerna
150	0,12	0,5	4	56	zmerna
220	0,12	0,5	4	56	zmerna
100	0,16	0,5	4	56	zmerna
150	0,16	0,5	4	56	zmerna
220	0,16	0,5	4	56	zmerna
100	0,08	0,25	8	56	zmerna
150	0,08	0,25	8	56	zmerna
220	0,08	0,25	8	56	zmerna
100	0,12	0,25	8	56	zmerna
150	0,12	0,25	8	56	zmerna
220	0,12	0,25	8	56	zmerna
100	0,16	0,25	8	56	zmerna
150	0,16	0,25	8	56	zmerna
220	0,16	0,25	8	56	zmerna
100	0,08	0,5	8	56	zmerna
150	0,08	0,5	8	56	zmerna
220	0,08	0,5	8	56	zmerna
100	0,12	0,5	8	56	zmerna
150	0,12	0,5	8	56	zmerna
220	0,12	0,5	8	56	zmerna
100	0,16	0,5	8	56	zmerna
150	0,16	0,5	8	56	zmerna
220	0,16	0,5	8	56	zmerna
100	0,08	0,25	2	63	zmerna
150	0,08	0,25	2	63	zmerna
220	0,08	0,25	2	63	zmerna
100	0,12	0,25	2	63	zmerna
150	0,12	0,25	2	63	zmerna
220	0,12	0,25	2	63	zmerna
100	0,16	0,25	2	63	zmerna
150	0,16	0,25	2	63	zmerna
220	0,16	0,25	2	63	zmerna
100	0,08	0,5	2	63	zmerna
150	0,08	0,5	2	63	zmerna
220	0,08	0,5	2	63	zmerna
100	0,12	0,5	2	63	zmerna
150	0,12	0,5	2	63	zmerna
220	0,12	0,5	2	63	zmerna
100	0,16	0,5	2	63	zmerna

Rezalna hitrost [m/min.]	Podajalna hitrost [mm/vrt]	Globina rezanja [mm]	Čas obdelave [s]	Trdota [HRC]	Obraba orodja
150	0,16	0,5	2	63	zmerna
220	0,16	0,5	2	63	zmerna
100	0,08	0,25	4	63	zmerna
150	0,08	0,25	4	63	zmerna
220	0,08	0,25	4	63	zmerna
100	0,12	0,25	4	63	zmerna
150	0,12	0,25	4	63	zmerna
220	0,12	0,25	4	63	zmerna
100	0,16	0,25	4	63	zmerna
150	0,16	0,25	4	63	zmerna
220	0,16	0,25	4	63	zmerna
100	0,08	0,5	4	63	zmerna
150	0,08	0,5	4	63	zmerna
220	0,08	0,5	4	63	zmerna
100	0,12	0,5	4	63	zmerna
150	0,12	0,5	4	63	zmerna
220	0,12	0,5	4	63	zmerna
100	0,16	0,5	4	63	zmerna
150	0,16	0,5	4	63	zmerna
220	0,16	0,5	4	63	zmerna
100	0,08	0,25	8	63	zmerna
150	0,08	0,25	8	63	zmerna
220	0,08	0,25	8	63	zmerna
100	0,12	0,25	8	63	zmerna
150	0,12	0,25	8	63	zmerna
220	0,12	0,25	8	63	zmerna
100	0,16	0,25	8	63	zmerna
150	0,16	0,25	8	63	zmerna
220	0,16	0,25	8	63	zmerna
100	0,08	0,5	8	63	zmerna
150	0,08	0,5	8	63	zmerna
220	0,08	0,5	8	63	zmerna
100	0,12	0,5	8	63	zmerna
150	0,12	0,5	8	63	zmerna
220	0,12	0,5	8	63	zmerna
100	0,16	0,5	8	63	zmerna
150	0,16	0,5	8	63	zmerna
220	0,16	0,5	8	63	zmerna
100	0,08	1,2	2	56	velika
150	0,08	1,2	2	56	velika
220	0,08	1,2	2	56	velika
100	0,12	1,2	2	56	velika
150	0,12	1,2	2	56	velika
220	0,12	1,2	2	56	velika
100	0,16	1,2	2	56	velika
150	0,16	1,2	2	56	velika
220	0,16	1,2	2	56	velika
100	0,08	1,2	4	56	velika
150	0,08	1,2	4	56	velika

Rezalna hitrost [m/min.]	Podajalna hitrost [mm/vrt]	Globina rezanja [mm]	Čas obdelave [s]	Trdota [HRC]	Obraba orodja
220	0,08	1,2	4	56	velika
100	0,12	1,2	4	56	velika
150	0,12	1,2	4	56	velika
220	0,12	1,2	4	56	velika
100	0,16	1,2	4	56	velika
150	0,16	1,2	4	56	velika
220	0,16	1,2	4	56	velika
100	0,08	1,2	8	56	velika
150	0,08	1,2	8	56	velika
220	0,08	1,2	8	56	velika
100	0,12	1,2	8	56	velika
150	0,12	1,2	8	56	velika
220	0,12	1,2	8	56	velika
100	0,16	1,2	8	56	velika
150	0,16	1,2	8	56	velika
220	0,16	1,2	8	56	velika
100	0,08	1,2	2	63	velika
150	0,08	1,2	2	63	velika
220	0,08	1,2	2	63	velika
100	0,12	1,2	2	63	velika
150	0,12	1,2	2	63	velika
220	0,12	1,2	2	63	velika
100	0,16	1,2	2	63	velika
150	0,16	1,2	2	63	velika
220	0,16	1,2	2	63	velika
100	0,08	1,2	4	63	velika
150	0,08	1,2	4	63	velika
220	0,08	1,2	4	63	velika
100	0,12	1,2	4	63	velika
150	0,12	1,2	4	63	velika
220	0,12	1,2	4	63	velika
100	0,16	1,2	4	63	velika
150	0,16	1,2	4	63	velika
220	0,16	1,2	4	63	velika
100	0,08	1,2	8	63	velika
150	0,08	1,2	8	63	velika
220	0,08	1,2	8	63	velika
100	0,12	1,2	8	63	velika
150	0,12	1,2	8	63	velika
220	0,12	1,2	8	63	velika
100	0,16	1,2	8	63	velika
150	0,16	1,2	8	63	velika
220	0,16	1,2	8	63	velika

STROJNO UČENJE ZA INŽENIRJE: KONCEPTI, PRIMERI IN UPORABA V OKOLJU MATLAB

JANEZ GOTLIH, MIRAN BREZOČNIK
Univerza v Mariboru, Fakulteta za strojništvo, Maribor, Slovenija
janez.gotlih@um.si, miran.brezocnik@um.si

Skripte obravnavajo strojno učenje z vidika uporabe v inženirstvu, pri čemer povezujejo temeljne koncepte s praktično uporabo v okolju MATLAB. Predstavljeni so štirje temeljni pristopi strojnega učenja: nadzorovano učenje, nenadzorovano učenje, učenje z okrepitvijo in prenosno učenje. Za vsak pristop so podani temeljni koncepti, konkretni primeri uporabe ter naloge za samostojno delo. Poseben poudarek je na uporabi orodij, kot so Regression Learner, Classification Learner, Deep Network Designer in Reinforcement Learning Designer, s pomočjo katerih študenti razvijajo modele na podatkih, ki izvirajo iz realnih inženirskih primerov. Ti so: obraba orodja, vibracije strojev, balansiranje sistemov in prepoznavanje predmetov. Skripta vključujejo tudi eksperimentalne podatkovne množice in praktične napotke za učenje, validacijo in izboljšavo modelov. Namenjena so študentom tehniških smeri ter vsem, ki želijo usvojiti uporabo metod strojnega učenja za reševanje konkretnih inženirskih problemov.

DOI
https://doi.org/10.18690/um.fs.10.2025

ISBN
978-961-299-078-7

Ključne besede:
strojno učenje,
nadzorovano učenje,
nenadzorovano učenje,
učenje z okrepitvijo,
prenosno učenje,
MATLAB,
inženirske aplikacije

Univerzitetna založba Univerze v Mariboru

DOI
https://doi.org/
10.18690/um.fs.10.2025

ISBN
978-961-299-078-7

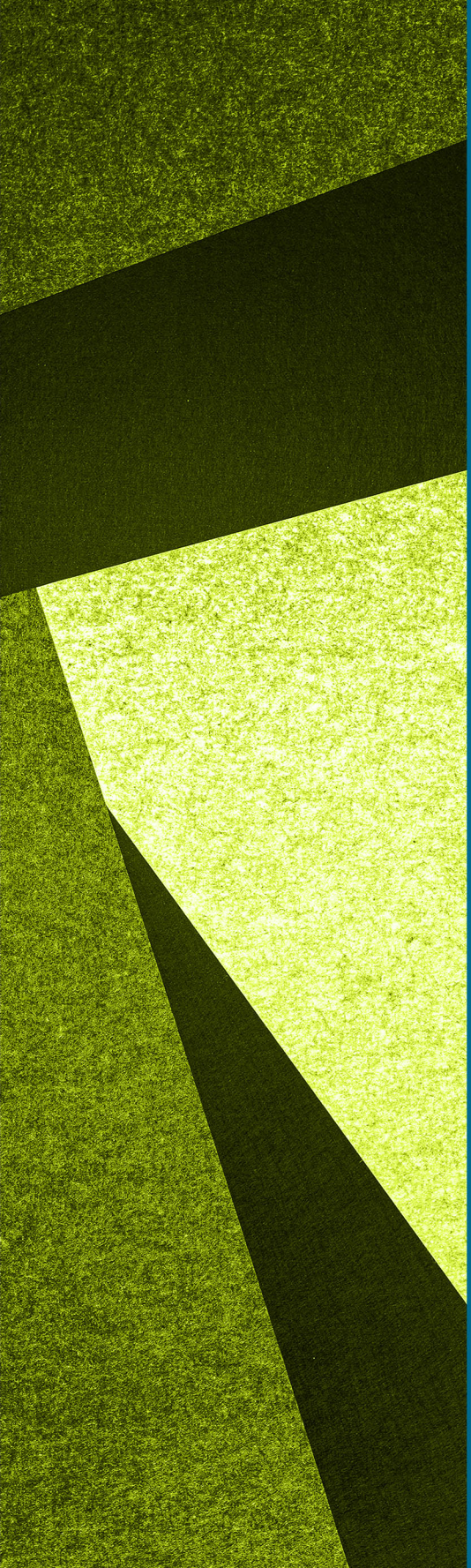
Machine Learning for Engineers: Concepts, Examples, and Applications in MATLAB

Janez Gotlih, Miran Brezočnik
University of Maribor, Faculty of Mechanical Engineering, Maribor, Slovenia
janez.gotlih@um.si, miran.brezocnik@um.si

Keywords:
machine learning,
supervised learning,
unsupervised learning,
reinforcement learning,
transfer learning,
MATLAB,
engineering applications

The book deala with machine learning from the perspective of its application in engineering, linking fundamental concepts with practical application in the MATLAB environment. Four basic approaches to machine learning are presented: supervised learning, unsupervised learning, reinforcement learning, and transfer learning. For each approach, basic concepts, specific use cases, and independent work assignments are provided. Special emphasis is placed on the use of tools such as Regression Learner, Classification Learner, Deep Network Designer, and Reinforcement Learning Designer, with which students develop models based on data derived from real engineering examples. These include tool wear, machine vibrations, system balancing, and object recognition. The scripts also include experimental data sets and practical guidelines for learning, validating, and improving models. They are intended for students of technical disciplines and anyone who wants to learn how to use machine learning methods to solve specific engineering problems.

University of Maribor Press



Univerza v Mariboru

Fakulteta za strojništvo

FS