Workplace and Process Safety in Next Generation Europe - Teaching for Learning Safety4EU

Project No. 101126832

Coordinator: Dr Milan Protić

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8: Mašinsko učenje u procesnoj bezbednosti — Pregled (16 h)

Co-funded by the European Union

Struktura kursa L8

- L8-1 Uvod u ML za procesnu bezbednost — pojmovi i ekosistem [2 h]
- L8-2 Priprema podataka i inženjerstvo karakteristika [2 h]
- L8-3 Nadzirano učenje — klasifikacija (SVM, stabla, ansambli) [3 h]
- L8-4 Nadzirano učenje — regresija i odabir modela [2 h]
- L8-5 Neuronske mreže i duboko učenje [3 h]
- L8-6 Nenadzirano učenje — klasterovanje i detekcija anomalija [2 h]
- L8-7 Vremenske serije i LSTM za predviđanje hazarda [2 h]

Fokus kursa i filozofija

- ML nije zamena za inženjera — to je alat koji pojačava sposobnost prosuđivanja.
- Primeri su iz industrijske prakse: senzori postrojenja, alarmni logovi i SEVESO podaci.
- Python: scikit-learn, Keras, pandas, NumPy i Matplotlib — otvoreni kod i besplatni alati.
- Maple: simbolička analiza, vizuelizacija i provera numeričkih rezultata iz Pythona.

Veza sa prethodnim lekcijama

- L5/L6: fizičko-hemijski modeli daju oznake i ulazne podatke za ML modele u L8.
- L8 izlaz: inteligentni alarmi, prediktivno održavanje i autonomna detekcija anomalija u SCADA sistemima.

Co-funded by the European Union

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-1 Uvod u ML za procesnu bezbednost — pojmovi i ekosistem

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-1: Šta je mašinsko učenje i gde se koristi u industriji?

Co-funded by the European Union

Definicija mašinskog učenja (ML)

- Skup metoda koje omogućavaju računarskim sistemima da uče iz podataka bez eksplicitnog programiranja.
- Arthur Samuel (1959): „polje koje daje računarima sposobnost učenja bez eksplicitnog programiranja".
- Razlika od klasičnog programiranja: pravila + podaci → odgovor; ML: podaci + odgovor → pravila.

Primene ML u procesnoj bezbednosti i industriji

- Detekcija anomalija u procesnim podacima (SCADA, DCS senzori) — rano upozorenje na kvarove.
- Prediktivno održavanje: predvideti kvar pumpe, kompresora ili ventila pre nego što se dogodi.
- Klasifikacija alarmnih stanja: razlikovanje lažnih alarma od pravih hazarda.
- Modelovanje disperzije: surogatni ML modeli umesto skupih CFD simulacija (L6).
- Optimizacija procesa: smanjenje energetske potrošnje uz održavanje bezbednosnih parametara.

Tri osnovne paradigme ML

- Nadzirano učenje (supervised): model se trenira na označenim primerima, X → y je poznato.
- Nenadzirano učenje (unsupervised): pronalaženje skrivene strukture u neoznačenim podacima.
- Učenje potkrepljenjem (reinforcement learning): agent uči kroz nagrade i kazne, npr. u kontroli procesa.

L8-1: ML ekosistem — Python biblioteke i alati

Co-funded by the European Union

Osnove Python ekosistema za ML

- NumPy: n-dimenzionalne matrice visokih performansi i matematičke operacije.
- Pandas: tabelarni podaci (DataFrame) — učitavanje, čišćenje i transformacija industrijskih podataka.
- Matplotlib / Seaborn: vizuelizacija podataka i rezultata modela.
- Scikit-learn (sklearn): standardna biblioteka za ML — klasifikacija, regresija, klasterovanje i CV.
- Keras / TensorFlow: duboko učenje — neuronske mreže, LSTM i CNN modeli.
- PyTorch: alternativa TensorFlow-u, posebno popularna u istraživanju.

Standardni tok rada ML projekta

- 1. Prikupljanje i razumevanje podataka (EDA — eksplorativna analiza podataka).
- 2. Čišćenje i priprema podataka — preprocessing i inženjerstvo karakteristika.
- 3. Odabir i treniranje modela.
- 4. Evaluacija modela — metrike i unakrsna validacija.
- 5. Fino podešavanje hiperparametara.
- 6. Uvođenje modela u rad i praćenje u produkciji (MLOps).

Co-funded by the European Union

Python — instalacija i provera okruženja

```
# pip install scikit-learn keras tensorflow pandas matplotlib seaborn
import sklearn; import numpy as np; import pandas as pd
print('sklearn version:', sklearn.__version__)
print('numpy version:', np.__version__)
```

L8-1: Matematičke osnove ML — linearna algebra i statistika

Co-funded by the European Union

Linearna algebra — ključni pojmovi

- Vektor $x = [x_1, x_2, \ldots, x_n]^T$: jedan primer, odnosno jedno merenje senzora.
- Matrica X oblika m × n: m primera i n karakteristika modela.
- Skalarni proizvod: $w^T x = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$ — osnova linearnih modela.
- Norma vektora: $||x||_2 = \sqrt{(x_1^2 + \ldots + x_n^2)}$ — mera veličine vektora.

Statističke osnove

- Srednja vrednost: $\mu = (1/n) \cdot \Sigma_i x_i$ — očekivana vrednost procesa.
- Standardna devijacija: $\sigma = \sqrt{((1/n) \cdot \Sigma_i (x_i - \mu)^2)}$ — varijabilnost podataka.
- Kovarijansa: $Cov(X,Y) = E[(X - \mu_x)(Y - \mu_y)]$ — linearna zavisnost dve promenljive.
- Pearsonov r: korelacioni koeficijent u intervalu [−1, 1] — jačina linearne veze.

L8-1: Matematičke osnove ML — linearna algebra i statistika

Co-funded by the European Union

Maple — simbolička linearna algebra

```
restart:
with(LinearAlgebra):
A := Matrix([[2,1,0],[1,3,1],[0,1,2]]):
# Sopstvene vrednosti (eigenvalues) — ključne za PCA:
Eigenvalues(A);
# Singularna dekompozicija (SVD):
SingularValues(A);
```

Python ekvivalent

```
import numpy as np
A = np.array([[2,1,0],[1,3,1],[0,1,2]])
vals, vecs = np.linalg.eig(A)
U, S, Vt = np.linalg.svd(A)
print('Sopstvene vrednosti:', vals.round(3))
```

L8-1: Overfitting i underfitting — osnovni problem ML

Co-funded by the European Union

Bias–variance kompromis

- Bias (pristrasnost): greška modela koji je suviše jednostavan — ne hvata pravu zavisnost (underfitting).
- Variance (varijansa): greška modela koji je suviše složen — uči šum umesto signala (overfitting).
- Ukupna greška: $E = \text{Bias}^2 + \text{Variance} + \text{nepovratni šum (irreducible noise)}$.
- Cilj: naći model optimalne složenosti — dovoljno fleksibilan, ali ne prekompleksan.

Underfitting (visok bias)

- Simptomi: greška na trening skupu je visoka i greška na test skupu je visoka.
- Uzroci: suviše jednostavan model, npr. linearni model za nelinearni problem.
- Rešenje: povećati složenost modela, dodati karakteristike i smanjiti regularizaciju.

Overfitting (visoka varijansa)

- Simptomi: greška na trening skupu je niska, a greška na test skupu znatno viša.
- Uzroci: model je suviše složen ili ima premalo podataka za treniranje.
- Rešenje: regularizacija (L1/L2), dropout kod neuronskih mreža, unakrsna validacija i više podataka.

L8-1: Overfitting i underfitting — osnovni problem ML

Co-funded by the European Union

Python — demonstracija overfittinga na primeru

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
np.random.seed(42)
x = np.linspace(0, 1, 50).reshape(-1,1)
y = np.sin(2*np.pi*x) + 0.2*np.random.randn(50,1)
for deg in [1, 3, 15]:  # underfitting, OK, overfitting
    poly = PolynomialFeatures(deg)
    Xp = poly.fit_transform(x)
    model = LinearRegression().fit(Xp, y)
    print(f'Stepen {deg}:
  MSE_train={mean_squared_error(y,model.predict(Xp)):.4f}')
```

L8-1: Podela podataka i unakrsna validacija

Co-funded by the European Union

Zašto deliti podatke?

- Cilj ML je model koji generalizuje na nove, neviđene podatke — ne samo da pamti trening skup.
- Trening skup: podaci na kojima model uči — obično 70–80% ukupnih podataka.
- Validacioni skup: podešavanje hiperparametara — obično 10–15%.
- Testni skup: finalna procena generalizacije — 10–20%; koristi se jednom.

k-puta unakrsna validacija (k-fold cross-validation)

- Podeli podatke na k jednakih delova; treniraj k puta, svaki put testirajući na drugom delu.
- CV skor = srednja vrednost test grešaka — robustna procena stvarnih performansi.
- k = 5 ili k = 10 su standardni izbori u industrijskoj praksi.
- Stratifikovana CV: za klasifikaciju — svaki fold ima isti odnos klasa kao ukupni skup.

L8-1: Podela podataka i unakrsna validacija

Co-funded by the European Union

Python — unakrsna validacija za procesne podatke

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
import numpy as np
# Simulovani podaci: 1000 merenja senzora, 2 klase (normalan/alarm)
X = np.random.randn(1000, 10)  # 10 senzora
y = (X[:,0] + X[:,2] > 0).astype(int)  # alarm ako suma senzora 1 i 3 >
  0
clf = RandomForestClassifier(n_estimators=100, random_state=42)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(clf, X, y, cv=cv, scoring='f1')
print(f'F1: {scores.mean():.3f} +/- {scores.std():.3f}')
```

L8-1: Metrike evaluacije modela — klasifikacija i regresija

Co-funded by the European Union

Matrica konfuzije

- TP (true positive): alarm je ispravno detektovan; FN (false negative): alarm je propušten — opasno.
- FP (false positive): lažni alarm; TN (true negative): ispravno prepoznato normalno stanje.
- U bezbednosti je FN mnogo opasniji od FP — model mora imati visok recall/odziv.

Metrike za klasifikaciju

- Accuracy: (TP + TN)/(TP + TN + FP + FN) — pouzdanost na uravnoteženim skupovima.
- Precision: TP/(TP + FP) — od svih alarma, koliko je pravih?
- Recall (sensitivity): TP/(TP + FN) — od svih pravih alarma, koliko je detektovano?
- F1-score: 2 · Precision · Recall/(Precision + Recall) — harmonijska sredina precision i recall.
- AUC-ROC: površina ispod ROC krive — mera sposobnosti razlikovanja klasa u intervalu [0.5, 1.0].

L8-1: Metrike evaluacije modela — klasifikacija i regresija

Co-funded by the European Union

Matrica konfuzije

- TP (true positive): alarm je ispravno detektovan; FN (false negative): alarm je propušten — opasno.
- FP (false positive): lažni alarm; TN (true negative): ispravno prepoznato normalno stanje.
- U bezbednosti je FN mnogo opasniji od FP — model mora imati visok recall/odziv.

Metrike za klasifikaciju

- Accuracy: (TP + TN)/(TP + TN + FP + FN) — pouzdanost na uravnoteženim skupovima.
- Precision: TP/(TP + FP) — od svih alarma, koliko je pravih?
- Recall (sensitivity): TP/(TP + FN) — od svih pravih alarma, koliko je detektovano?
- F1-score: 2 · Precision · Recall/(Precision + Recall) — harmonijska sredina precision i recall.
- AUC-ROC: površina ispod ROC krive — mera sposobnosti razlikovanja klasa u intervalu [0.5, 1.0].

L8-1: Metrike evaluacije modela — klasifikacija i regresija

Co-funded by the European Union

Metrike za regresiju

- MAE: $(1/n) \cdot \Sigma_i |y_i - \hat{y}_i|$ — prosečna apsolutna greška.
- RMSE: $\sqrt{((1/n) \cdot \Sigma_i (y_i - \hat{y}_i)^2)}$ — jače kažnjava velike greške od MAE.
- R^2: $1 - SS_{res}/SS_{tot}$ — udeo objašnjene varijanse; 1.0 je savršeno.

Python — sve metrike odjednom

```
from sklearn.metrics import classification_report, confusion_matrix,
  roc_auc_score
print(classification_report(y_test, y_pred))
print('AUC-ROC:', roc_auc_score(y_test, y_prob[:,1]))
```

L8-1: Tipovi podataka u industrijskim postrojenjima

Co-funded by the European Union

Strukturirani procesni podaci (SCADA/DCS)

- Vremenski označeni signali senzora: temperatura, pritisak, protok, nivo i koncentracija.
- Frekvencija uzorkovanja: 1 Hz do 100 Hz za procesne varijable; 1/60 Hz za energetiku.
- Tipična postrojenja: DCS sistemi generišu milione tačaka podataka dnevno.
- OSIsoft PI, Honeywell PHD i Siemens PCS 7 — industrijski SCADA sistemi sa ML integracijama.

Alarmni podaci

- Alarmni log: oznaka taga, vreme alarma, prioritet, trajanje i akcija operatera.
- Poplava alarma: više od 30 alarma/min — operater ne može da reaguje na sve.
- ML za upravljanje alarmima: razlikovanje važnih od lažnih alarma i klasterovanje srodnih događaja.

Nestrukturirani podaci

- Izveštaji o nesrećama: NLP za ekstrakciju uzroka i obrazaca iz teksta.
- Slike i video: računarski vid za detekciju curenja, požara i neispravne opreme.
- Audio: akustična emisija za detekciju pukotina u cevovodima — ultrazvučni senzori + ML.

Co-funded by the European Union

Python — učitavanje i osnovna analiza vremenskih serija senzora

```
import pandas as pd; import matplotlib.pyplot as plt
df = pd.read_csv('scada_data.csv', parse_dates=['timestamp'],
  index_col='timestamp')
print(df.describe())  # statistike svih senzora
df[['T_reaktor','P_reaktor']].plot(subplots=True, figsize=(12,6))
plt.suptitle('SCADA vremenski nizovi — reaktor')
```

L8-1: Primer skupa podataka — industrijska procesna bezbednost

Co-funded by the European Union

Tennessee Eastman Process (TEP) — standardni referentni skup

- Simulirani hemijski proces sa 52 merenja i 12 različitih tipova kvarova.
- Jedan od najkorišćenijih referentnih skupova za detekciju anomalija i klasifikaciju kvarova u procesnoj industriji.
- Podaci su dostupni preko UCI ML Repository i Kaggle platforme.

SWAT (Secure Water Treatment) skup podataka

- Realno industrijsko vodovodno postrojenje sa simuliranim sajber-fizičkim napadima.
- 143 senzora i aktuatora, 36 normalnih i 36 napadnutih epizoda — iTrust Singapore.

Gas Sensor Array skup podataka (UCI)

- 16 metal-oksidnih gasnih senzora — metan/etanol smeše u različitim uslovima.
- Primena: detekcija ATEX zona i rano upozoravanje na curenje gasa u postrojenjima.

L8-1: Primer skupa podataka — industrijska procesna bezbednost

Co-funded by the European Union

Python — učitavanje TEP skupa podataka

```
import pandas as pd; import numpy as np
# TEP podaci: prvih 22 kolone = kontinualna merenja (XMEAS)
# Kolone 23-41 = manipulisane varijable (XMV)
# Poslednja kolona = fault ID (0 = normalan rad)
url = 'https://raw.githubusercontent.com/YKatser/CPDE-
  TEP/master/data/d00.dat'
df = pd.read_csv(url, sep='\t', header=None)
print(f'Oblik skupa: {df.shape}')  # (500, 52)
print(df.describe().round(2))
```

Co-funded by the European Union

Maple — osnovna statistička analiza mernih podataka

```
restart:
with(Statistics):
# Simulovani podaci temperature reaktora (normalan rad vs. kvar):
T_normal := Sample(Normal(80, 3), 500):   # normalan rad: T ~ N(80, 3)
T_kvar   := Sample(Normal(95, 8), 100):   # kvar: T ~ N(95, 8)
T_sve    := [op(T_normal), op(T_kvar)]:

# Deskriptivna statistika:
printf('Srednja T_normal = %.2f, sigma = %.2f\n',
       Mean(T_normal), StandardDeviation(T_normal)):

# Histogram i gustina verovatnoće:
with(plots):
h1 := Histogram(T_normal, color=blue, transparency=0.4):
h2 := Histogram(T_kvar, color=red, transparency=0.4):
display([h1, h2], title='Raspodela temperature: normalan vs. kvar'):

# T-test za razliku sredina (statistička značajnost):
TwoSampleTTest(T_normal, T_kvar, level=0.05):
```

L8-1: Upravljanje podacima i etika ML u bezbednosti

Co-funded by the European Union

Kvalitet podataka — GIGO princip

- GIGO (garbage in, garbage out): ML model je tačan onoliko koliko su dobri ulazni podaci.
- Problemi u industrijskim podacima: nedostajuće vrednosti, kvarovi senzora, kalibracija i drift.
- Pristrasnost podataka: ako kvarovi nisu zastupljeni, model ih neće naučiti.
- Princip: uvek razumeti fizičku osnovu podataka pre primene ML — znanje domena je ključno.

Neravnoteža klasa u bezbednosti

- Normalan rad: 99.9% vremena; kvar/alarm: 0.1% — ekstremno neuravnoteženi podaci.
- Rešenja: SMOTE, class_weight i undersampling, uz pažljivu validaciju rezultata.

Etika i odgovornost ML u industrijskoj bezbednosti

- Objašnjivost: operater mora razumeti zašto model daje alarm — crna kutija je problem.
- SHAP (Shapley Additive Explanations): kvantifikuje doprinos svakog ulaza odluci modela.
- LIME: lokalna objašnjivost za jedan primer i jednu odluku modela.
- Regulativa: EU AI Act (2024) — AI sistemi visokog rizika u bezbednosti zahtevaju objašnjivost.

Co-funded by the European Union

Python — SMOTE za neuravnotežene klase

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42, k_neighbors=5)
X_res, y_res = sm.fit_resample(X_train, y_train)
print(f'Pre: {np.bincount(y_train)} | Posle: {np.bincount(y_res)}')
```

L8-1: Sistemi za praćenje procesa — DCS i SCADA integracija ML

Co-funded by the European Union

Arhitektura industrijskog sistema za praćenje

- Nivo 0: senzori i aktuatori — fizički uređaji u postrojenju.
- Nivo 1: PLC / DCS — kontrola u realnom vremenu.
- Nivo 2: SCADA — nadzor, prikupljanje podataka i alarmi.
- Nivo 3: MES — planiranje i optimizacija proizvodnje.
- Nivo 4: ERP — poslovni nivo i integracija sa upravljanjem resursima.

Gde se ML uključuje?

- Edge ML (nivo 1): ML direktno na PLC/DCS čipovima; latencija < 1 ms, važno za ESD.
- Fog/SCADA ML (nivo 2): analize na lokalnom serveru postrojenja — minutna latencija.
- Cloud ML (nivo 3+): batch analize, treniranje modela i poređenje više postrojenja.
- OPC-UA protokol: standardni industrijski protokol za razmenu podataka između nivoa.

Co-funded by the European Union

Python — simulacija SCADA toka sa detekcijom anomalija

```
import numpy as np; from collections import deque
# Klizeći prozor za detekciju anomalija (Z-score metoda):
window = deque(maxlen=100)  # poslednjih 100 merenja
def detektuj_anomaliju(nova_vrednost, prag=3.0):
    window.append(nova_vrednost)
    if len(window) < 20: return False
    z = (nova_vrednost - np.mean(window)) / (np.std(window) + 1e-9)
    return abs(z) > prag
```

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-2 Priprema podataka i inženjerstvo karakteristika

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-2: Priprema podataka — osnova svakog uspešnog ML projekta

Co-funded by the European Union

Zašto je priprema podataka toliko važna?

- U praksi se često najveći deo vremena troši na pripremu podataka, a manji deo na samo modelovanje.
- Lošiji podaci + bolji algoritam < bolji podaci + jednostavniji algoritam.
- U industrijskim postrojenjima javljaju se otkazi senzora, zasićenje opsega, drift kalibracije i komunikacioni prekidi.

Koraci pripreme podataka

- Učitavanje i inspekcija: oblik, tipovi podataka, statistike, nedostajuće vrednosti i duplikati.
- Rukovanje nedostajućim vrednostima: brisanje, imputacija srednjom vrednošću, medijanom ili modelom.
- Detekcija i tretman odstupajućih vrednosti: IQR metoda, z-score i Isolation Forest.
- Skaliranje i normalizacija: StandardScaler, MinMaxScaler i RobustScaler.
- Enkodiranje kategorijskih promenljivih: one-hot, label encoding i target encoding.
- Inženjerstvo karakteristika: kreiranje novih informativnih promenljivih iz postojećih merenja.

L8-2: Priprema podataka — osnova svakog uspešnog ML projekta

Co-funded by the European Union

Python — kompletna EDA analiza procesnih podataka

```
import pandas as pd; import seaborn as sns; import matplotlib.pyplot as
  plt
df = pd.read_csv('reactor_sensors.csv', parse_dates=['ts'],
  index_col='ts')
print('Oblik:', df.shape)
print('Nedostajuće vrednosti:\n', df.isnull().sum())
print('Statistike:\n', df.describe().T)
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Korelaciona matrica senzora')
```

L8-2: Rukovanje nedostajućim vrednostima u vremenskim serijama

Co-funded by the European Union

Uzroci nedostajućih podataka u industrijskim merenjima

- Kvar senzora: senzor ne šalje signal tokom servisa ili otkaza.
- Komunikacioni prekid: OPC-UA/MODBUS greška — kratki ispadi od sekundi do minuta.
- Zasićenje mernog opsega: senzor vraća saturisanu vrednost, tipično 0 ili maksimum, umesto NaN.
- Održavanje: senzori su isključeni tokom planiranog održavanja, često duže vreme.

Strategije popunjavanja vrednosti

- Forward fill (ffill): prenosi poslednju validnu vrednost napred — pogodno za kratke ispade (< 5 min).
- Linearna interpolacija: linearan prelaz između dve poznate tačke — dobra za stabilne procese.
- Splajn interpolacija: glađi prelaz — korisna za procese sa nelinearnim tendencijama.
- ML imputacija: KNN ili Random Forest uče vrednost iz ostalih senzora — najtačnije, ali sporije.

L8-2: Rukovanje nedostajućim vrednostima u vremenskim serijama

Co-funded by the European Union

Python — imputacija vremenskih serija

```
import pandas as pd; import numpy as np
from sklearn.impute import KNNImputer
# Simuliramo 5% nedostajućih vrednosti:
mask = np.random.rand(*df.shape) < 0.05
df_missing = df.copy(); df_missing[mask] = np.nan
# Forward fill za kratke ispade (< 3 uzorka):
df_ff = df_missing.ffill(limit=3)
# KNN imputacija za duže ispade:
imp = KNNImputer(n_neighbors=5)
df_knn = pd.DataFrame(imp.fit_transform(df_missing), columns=df.columns)
mae_ff  = np.nanmean(np.abs(df_ff.values - df.values))
mae_knn = np.nanmean(np.abs(df_knn.values - df.values))
print(f'MAE ffill: {mae_ff:.4f} | MAE KNN: {mae_knn:.4f}')
```

L8-2: Detekcija i tretman odstupajućih vrednosti

Co-funded by the European Union

Zašto su odstupajuće vrednosti važne u bezbednosti?

- Odstupanje može biti senzorska greška koju treba ignorisati ili stvarni fizički događaj koji je alarm.
- Ključno je razlikovati grešku merenja od stvarnog procesa — ML detekcija anomalija je koristan alat.

IQR metoda

- IQR = $Q_3 - Q_1$ — razlika 75. i 25. percentila.
- Granice: donja = $Q_1 - 1.5 \cdot IQR$; gornja = $Q_3 + 1.5 \cdot IQR$.
- Vrednosti van granica su potencijalni autlajeri, ali IQR je naivan za vremenske serije.

z-score metoda

- $z = (x - \mu)/\sigma$; ako je $|z| > 3$, vrednost se tretira kao mogući autlajer.
- Metoda je osetljiva na same autlajere, jer oni menjaju srednju vrednost i standardnu devijaciju.
- Za procesne podatke često je bolji robust z-score zasnovan na medijani i MAD meri.

L8-2: Detekcija i tretman odstupajućih vrednosti

Co-funded by the European Union

Python — robusna detekcija autlajera u procesnim podacima

```
import numpy as np; import pandas as pd
def robust_zscore(x):
    med = np.median(x)
    mad = np.median(np.abs(x - med))  # Median Absolute Deviation
    return 0.6745 * (x - med) / (mad + 1e-9)
z = robust_zscore(df['T_reaktor'].values)
autlaieri = np.where(np.abs(z) > 3.5)[0]
print(f'Detektovano {len(autlaieri)} autlajera od {len(z)} uzoraka')
```

L8-2: Skaliranje i normalizacija karakteristika

Co-funded by the European Union

Zašto je skaliranje važno?

- Senzori imaju različite opsege: T = 20 – 350 °C, P = 0 – 100 bar, protok = 0 – 5000 L/h.
- Algoritmi zasnovani na udaljenosti (KNN, SVM) i gradijentnom spustu (NN) osetljivi su na razmere.
- Bez skaliranja karakteristika sa većim opsegom dominira i može potisnuti ostale ulaze.

StandardScaler — z-score normalizacija

- $x' = (x - \mu)/\sigma$; rezultat ima srednju vrednost 0 i standardnu devijaciju 1.
- Pretpostavlja približno normalnu raspodelu — nije idealan za jako iskošene ili bimodalne podatke.

MinMaxScaler

- $x' = (x - \min(x))/(\max(x) - \min(x))$; rezultat je u opsegu [0, 1].
- Osetljiv je na autlajere — jedna ekstremna vrednost može sabiti ceo skup u mali opseg.

L8-2: Skaliranje i normalizacija karakteristika

Co-funded by the European Union

RobustScaler — preporuka za procesne podatke

- x′ = (x − median)/IQR — robustan na autlajere i pogodan za SCADA podatke.

Python — skaliranje u sklearn Pipeline

```
from sklearn.preprocessing import RobustScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import GradientBoostingClassifier
pipe = Pipeline([
    ('scaler', RobustScaler()),
    ('clf', GradientBoostingClassifier(n_estimators=100))
])
pipe.fit(X_train, y_train)
print('Accuracy:', pipe.score(X_test, y_test))
```

L8-2: Inženjerstvo karakteristika za procesne podatke

Co-funded by the European Union

Šta je inženjerstvo karakteristika?

- Kreiranje novih informativnih ulaznih promenljivih iz sirovih merenja koje bolje opisuju fiziku procesa.
- Dobre karakteristike su često važnije od složenijeg algoritma — znanje domena je ključno.

Karakteristike u vremenskom domenu

- Klizeća sredina: $\mu_n = (x_{n-}w_{+1} + \ldots + x_n)/w$ — filtriranje šuma i praćenje trenda.
- Klizeća standardna devijacija: σ_n — mera varijabilnosti u prozoru.
- Brzina promene: $dx/dt = (x_n - x_{n-1})/\Delta t$ — mogući preduvesnik kvara.
- Energija signala: $E_n = \text{mean}(x_i^2)$ u prozoru — ukupna energija signala.

Karakteristike u frekvencijskom domenu

- FFT: pretvara vremenski signal u frekvencijski spektar.
- Dominantna frekvencija: frekvencija sa najvećom amplitudom — indikator oscilacija ili vibracija.
- Spektralna entropija: mera neuređenosti u frekvencijskom spektru — može rasti pred kvarom.

Co-funded by the European Union

Python — automatizovana ekstrakcija karakteristika

```
import numpy as np; import pandas as pd
def extract_features(signal, w=50):
    return {
        'mean':   np.mean(signal[-w:]),
        'std':    np.std(signal[-w:]),
        'roc':    (signal[-1]-signal[-2]) if len(signal)>1 else 0,
        'energy': np.mean(signal[-w:]**2),
        'dom_freq': np.argmax(np.abs(np.fft.rfft(signal[-w:])))
    }
```

L8-2: Smanjenje dimenzionalnosti — PCA i t-SNE

Co-funded by the European Union

Prokletstvo dimenzionalnosti

- Sa povećanjem broja karakteristika potrebno je eksponencijalno više podataka za treniranje.
- Postrojenja imaju stotine senzora; 500 senzora × 10 vremenskih karakteristika = 5000 dimenzija.
- Mnoge dimenzije su korelisane i nose redundantnu informaciju.

PCA — analiza glavnih komponenti

- PCA pronalazi ortogonalne pravce sa maksimalnom varijansom podataka.
- PC1 objašnjava najviše varijanse, PC2 sledeću itd. — komponente su sortirane po važnosti.
- Obično biramo prvih k komponenti koje zajedno objašnjavaju ≥ 95% ukupne varijanse.

Co-funded by the European Union

Python — PCA na TEP podacima

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt; import numpy as np
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)
pca = PCA(n_components=0.95)  # zadrži 95% varijanse
X_pca = pca.fit_transform(X_scaled)
print(f'Smanji: {X_train.shape[1]} -> {X_pca.shape[1]} dimenzija')
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.axhline(0.95, color='r', ls='--', label='95% praga')
plt.xlabel('Broj PC'); plt.ylabel('Kumulativno objašnjena varijansa')
from sklearn.manifold import TSNE
X_tsne = TSNE(n_components=2, random_state=42).fit_transform(X_scaled)
```

Co-funded by the European Union

Maple — PCA i korelaciona analiza

```
restart:
with(Statistics): with(LinearAlgebra): with(plots):
# Simulovani podaci 3 senzora sa korelacijom:
n := 200:
T := Sample(Normal(80, 5), n):
P := 2.5*T + Sample(Normal(0, 2), n):  # P korelisan sa T
F := Sample(Normal(100, 10), n):       # F nezavisan
# Matrica podataka:
X := Transpose(Matrix([T, P, F])):
# Centriranje:
Xc := X - Matrix(n, 3, (i,j) -> Mean(Column(X, j))):
# Kovarijansna matrica:
C := Transpose(Xc).Xc / (n-1):
printf('Kovarijansna matrica:\n'):
print(C):
# SVD za PCA:
U, S, Vt := SingularValueDecomposition(Xc):
# Udeo objašnjene varijanse PC1:
var_PC1 := S[1,1]^2 / add(S[i,i]^2, i=1..3):
printf('PC1 objašnjava %.1f%% varijanse\n', 100*var_PC1):
```

L8-2: Odabir važnih karakteristika

Co-funded by the European Union

Zašto birati karakteristike umesto koristiti sve?

- Manje karakteristika znači brži model, manji rizik od overfittinga i lakšu interpretaciju.
- Nebitne karakteristike dodaju šum koji otežava učenje, posebno kod malih skupova podataka.

Metode odabira karakteristika

- Filter metode: procenjuju karakteristike bez ML modela — korelacija, hi-kvadrat i mutual information.
- Wrapper metode: iterativno dodaju ili uklanjaju karakteristike i mere performanse (RFE).
- Embedded metode: važnost karakteristika ugrađena je u model — Decision Tree i Lasso regularizacija.

SHAP za važnost karakteristika u modelima crne kutije

- SHAP vrednost pokazuje koliko svaka karakteristika doprinosi predikciji za konkretan primer.
- Pozitivne SHAP vrednosti povećavaju predikciju, a negativne je smanjuju.
- SHAP summary plot daje pregled važnosti svih karakteristika za ceo skup podataka.

L8-2: Odabir važnih karakteristika

Co-funded by the European Union

Python — RFE i SHAP važnost karakteristika

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
import shap
# RFE – izabrati top 10 karakteristika:
clf = RandomForestClassifier(n_estimators=100, random_state=42)
rfe = RFE(clf, n_features_to_select=10)
rfe.fit(X_train, y_train)
print('Izabrane karakteristike:', np.where(rfe.support_)[0])
# SHAP summary:
clf.fit(X_train, y_train)
explainer = shap.TreeExplainer(clf)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values[1], X_test, feature_names=feature_names)
```

L8-2: Pipeline i automatizacija ML toka za industrijska postrojenja

Co-funded by the European Union

sklearn Pipeline — jedinstven lanac obrade i modela

- Pipeline je sekvencijalan lanac transformacija i finalnog estimatora u jednom objektu.
- Prednosti: nema curenja informacija, jer se skaliranje uči samo na trening skupu, a ne na test skupu.
- Omogućava jednostavnu unakrsnu validaciju, podešavanje hiperparametara i uvođenje celog toka u rad.

Python — kompletan ML Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.impute import KNNImputer
from sklearn.decomposition import PCA
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
```

L8-2: Pipeline i automatizacija ML toka za industrijska postrojenja

Co-funded by the European Union

Python — kompletan ML Pipeline…nastavak

```
pipe = Pipeline([
    ('imputer', KNNImputer(n_neighbors=5)),
    ('scaler',  RobustScaler()),
    ('pca',     PCA(n_components=0.95)),
    ('clf',     GradientBoostingClassifier())
])
param_grid = {
    'clf__n_estimators': [50, 100, 200],
    'clf__max_depth': [3, 5, 7],
    'clf__learning_rate': [0.05, 0.1, 0.2]
}
gs = GridSearchCV(pipe, param_grid, cv=5, scoring='f1', n_jobs=-1)
gs.fit(X_train, y_train)
print('Najbolji F1:', gs.best_score_, '| Parametri:', gs.best_params_)
```

Co-funded by the European Union

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-3 Nadzirano učenje — klasifikacija (SVM, stabla, ansambli)

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-3: Nadzirano učenje — uvod u klasifikaciju

Co-funded by the European Union

Klasifikacioni problem

- Cilj: naučiti funkciju f: X → y, gde y pripada skupu diskretnih oznaka, odnosno klasa.
- Binarna klasifikacija: y ∈ {0, 1} — normalan/alarm, bezbedan/opasan, kvar/bez kvara.
- Višeklasna klasifikacija: y ∈ {0, 1, 2, …, k} — tip kvara, npr. TEP sa više klasa.

Primene u procesnoj bezbednosti

- Detekcija kvarova senzora: normalan rad, zasićenje, brzi kvar ili spori drift.
- Klasifikacija alarmnih stanja: hladan start, stabilan rad, prekoračenje parametara ili hitna situacija.
- Klasifikacija tipa incidenta: curenje, požar, eksplozija ili toksično oslobađanje.

Algoritmi koje ćemo pokriti

- Logistička regresija: linearni model za klasifikaciju — osnovan, brz i objašnjiv.
- k-nearest neighbors (kNN): odluka na osnovu najsličnijih primera u bazi.
- Stablo odlučivanja: intuitivno, objašnjivo i lako za otklanjanje grešaka.
- Random Forest: ansambl stabala — jedan od najrobusnijih algoritama za industrijske podatke.
- Gradient Boosting (XGBoost, LightGBM): savremeni standard za tabelarne podatke.
- Support Vector Machine (SVM): efikasan za visoke dimenzije i manje skupove podataka.

Co-funded by the European Union

Logistička regresija — matematička osnova

- Linearna kombinacija karakteristika: $z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = w^Tx$.
- Sigmoid funkcija: $\sigma(z) = 1/(1 + \exp(-z))$ — pretvara z u verovatnoću u intervalu [0, 1].
- $P(y = 1 \mid x) = \sigma(w^Tx)$; predikcija: $\hat{y} = 1$ ako je $P > 0.5$, inače 0.

Trening — minimizacija log-loss funkcije

- $L(w) = -(1/m) \cdot \Sigma[y \cdot \log(p) + (1 - y) \cdot \log(1 - p)]$.
- Regularizacija: L_2 (Ridge): $L \mathrel{+}= \lambda||w||_2^2$; L_1 (Lasso): $L \mathrel{+}= \lambda||w||_1$.
- Gradijentni spust: $w = w - \alpha \cdot \nabla L(w)$ — iterativno ažuriranje težina.

Co-funded by the European Union

Primer — klasifikacija alarmnih stanja reaktora (Python)

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.preprocessing import StandardScaler
import numpy as np
scaler = StandardScaler()
X_tr_s = scaler.fit_transform(X_train)
X_te_s = scaler.transform(X_test)
lr = LogisticRegression(C=1.0, max_iter=1000, class_weight='balanced')
lr.fit(X_tr_s, y_train)
y_pred = lr.predict(X_te_s)
print(classification_report(y_test, y_pred, target_names=['Normalan','Alarm']))
print('AUC:', roc_auc_score(y_test, lr.predict_proba(X_te_s)[:,1]))
```

L8-3: Stablo odlučivanja

Co-funded by the European Union

Princip stabla odlučivanja

- Rekurzivno deljenje prostora karakteristika na osnovu najinformativnijeg pravila, odnosno split-a.
- Svaki unutrašnji čvor ima pravilo tipa „temperatura reaktora > 120 °C?“; listovi daju klasu ili verovatnoću.
- Kriterijumi za split: Ginijeva nečistoća, informacioni dobitak (entropija) i hi-kvadrat.
- Gini = $1 - \Sigma p_i^2$, gde je p_i udeo klase i u čvoru; Gini = 0 znači čist čvor.

Prednosti i mane

- Prednosti: intuitivno i objašnjivo — može se prikazati kao dijagram odlučivanja.
- Prednosti: automatska selekcija karakteristika, radi bez skaliranja i može tolerisati nedostajuće vrednosti.
- Mane: veoma sklono preprilagođavanju — duboka stabla pamte trening podatke.
- Rešenje: max_depth, min_samples_leaf i unakrsna validacija.

Co-funded by the European Union

Python — stablo odlučivanja za klasifikaciju alarmnih stanja TEP

```
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
import matplotlib.pyplot as plt
dt = DecisionTreeClassifier(max_depth=5, min_samples_leaf=20,
                            class_weight='balanced', random_state=42)
dt.fit(X_train, y_train)
# Tekstualni prikaz stabla:
print(export_text(dt, feature_names=feature_names, max_depth=3))
# Graficki prikaz stabla:
fig, ax = plt.subplots(figsize=(20,8))
plot_tree(dt, feature_names=feature_names, class_names=['Normalan','Alarm'],
          filled=True, fontsize=8, ax=ax)
print('Test Accuracy:', dt.score(X_test, y_test))
```

L8-3: Random Forest — ansambl stabala

Co-funded by the European Union

Ideja ansambla

- Kombinovati predviđanja više slabih modela da se dobije jedan jak model.
- Bagging (bootstrap agregacija): trenirati k stabala na različitim bootstrap podskupovima.
- Random Forest dodaje slučajnost pri izboru karakteristika za svaki split — smanjuje korelaciju stabala.
- Finalna predikcija: glasanje većine za klasifikaciju ili srednja vrednost za regresiju.

Prednosti Random Forest algoritma u industriji

- Otporan je na odstupajuće vrednosti i šum zbog glasanja većeg broja stabala.
- Daje automatsku važnost karakteristika (feature importance), što pomaže interpretaciji.
- Dobro radi bez velikog podešavanja hiperparametara; $n_{estimators} = 100$ je dobra početna vrednost.
- Moguća je Out-of-Bag (OOB) procena greške bez posebnog validacionog skupa.

Co-funded by the European Union

Python — Random Forest za klasifikaciju kvarova TEP postrojenja

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt; import numpy as np
rf = RandomForestClassifier(n_estimators=200, max_depth=15,
                            min_samples_leaf=5, class_weight='balanced',
                            oob_score=True, n_jobs=-1, random_state=42)
rf.fit(X_train, y_train)
print('OOB Score:', rf.oob_score_)
print('Test Accuracy:', rf.score(X_test, y_test))
# Feature importance:
importances = pd.Series(rf.feature_importances_, index=feature_names)
importances.sort_values().tail(10).plot(kind='barh')
plt.title('Top 10 features – Random Forest'); plt.tight_layout()
```

Co-funded by the European Union

Princip gradijentnog boosting-a

- Boosting: stabla se treniraju sekvencijalno — svako naredno popravlja grešku prethodnog.
- Novo stablo uči na rezidualima prethodnih stabala — gradijentni spust u prostoru funkcija.
- Rezultat je zbir k stabala: $F(x) = T_1(x) + T_2(x) + \ldots + T_k(x)$.
- XGBoost (Chen & Guestrin, 2016): veoma uspešan algoritam na Kaggle takmičenjima i tabelarnim podacima.
- Regularizovani GBM dodaje $L_1 + L_2$ penalizaciju na listove stabla, što smanjuje preprilagođavanje.
- Efikasna implementacija: moguće paralelno treniranje, pruning i optimizovan rad sa memorijom.
- Automatski rukuje nedostajućim vrednostima — često nije potrebna prethodna imputacija.
- LightGBM je Microsoftova alternativa, brža od XGBoost-a za velike skupove podataka.

Co-funded by the European Union

Python — XGBoost za klasifikaciju kvarova

```
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
xgb = XGBClassifier(
    n_estimators=300, max_depth=6, learning_rate=0.05,
    subsample=0.8, colsample_bytree=0.8,
    scale_pos_weight=10,  # za neravnotezene klase
    use_label_encoder=False, eval_metric='logloss',
    random_state=42, n_jobs=-1
)
xgb.fit(X_train, y_train,
        eval_set=[(X_test, y_test)], early_stopping_rounds=20,
        verbose=False)
print(classification_report(y_test, xgb.predict(X_test)))
```

L8-3: Mašina potpornih vektora (SVM)

Co-funded by the European Union

SVM — ideja maksimalnog marginalnog razdvajanja

- Pronaći hiperravan koja maksimizuje marginu između klasa u prostoru karakteristika.
- Potporni vektori su tačke najbliže hiperravni — one određuju marginu.
- Hard-margin SVM važi za linearno separabilne podatke, a soft-margin SVM koristi C parametar za toleranciju grešaka.

Kernel trik — SVM za nelinearne probleme

- Podaci se implicitno preslikavaju u višedimenzionalni prostor u kome mogu biti linearno separabilni.
- RBF kernel: $K(x, z) = \exp(-\gamma||x - z||^2)$ — pogodan za kružne i glatke granice odlučivanja.
- Polinomni kernel: $K(x, z) = (x^T z + c)^d$ — daje polinomnu granicu odlučivanja.
- Hiperparametri: C (kompromis margina/greška) i γ (širina RBF kernela) — podešavaju se unakrsnom validacijom.

Co-funded by the European Union

Python — SVM sa RBF kernelom i podešavanjem hiperparametara

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_tr_s = scaler.fit_transform(X_train)
X_te_s = scaler.transform(X_test)
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': ['scale', 0.01, 0.1]}
svm = GridSearchCV(SVC(kernel='rbf', probability=True, class_weight='balanced'),
                   param_grid, cv=5, scoring='f1', n_jobs=-1)
svm.fit(X_tr_s, y_train)
print('Najbol params:', svm.best_params_)
print('Test F1:', svm.score(X_te_s, y_test))
```

L8-3: Kompletna studija slučaja — klasifikacija kvarova pumpe

Co-funded by the European Union

Scenario: prediktivno održavanje centrifugalne pumpe

- Podaci: 6 senzora — temperatura ležaja, temperatura ulja, vibracija X/Y, protok i pritisak; 10000 uzoraka.
- Četiri klase: 0 = normalan rad, 1 = kvar ležaja, 2 = kavitacija, 3 = neizbalansiranost rotora.

Opis skupa podataka i EDA

```
import pandas as pd; import seaborn as sns; import matplotlib.pyplot as plt
df = pd.read_csv('pump_data.csv')
print('Raspodela klasa:
', df['fault'].value_counts())
sns.pairplot(df, hue='fault', vars=['T_lezaj','vibracija_X','protok'],
diag_kind='kde')
```

Co-funded by the European Union

Treniranje i poređenje više modela

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC; from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score; from sklearn.pipeline
import Pipeline
from sklearn.preprocessing import RobustScaler
modeli = {'DT': DecisionTreeClassifier(max_depth=8),
          'RF': RandomForestClassifier(n_estimators=100),
          'GB': GradientBoostingClassifier(n_estimators=100),
          'SVM':SVC(kernel='rbf', class_weight='balanced')}
for naziv, model in modeli.items():
    pipe = Pipeline([('sc', RobustScaler()), ('m', model)])
    f1 = cross_val_score(pipe, X, y, cv=5, scoring='f1_macro')
    print(f'{naziv}: F1={f1.mean():.3f}+/-{f1.std():.3f}')
```

L8-3: ROC kriva i Precision–Recall kriva za bezbednosne sisteme

Co-funded by the European Union

ROC kriva

- Grafikon TPR (recall) u odnosu na FPR (false positive rate) za sve pragove odlučivanja u intervalu [0, 1].
- Idealan model je u tački (0, 1), u gornjem levom uglu; nasumičan model prati dijagonalu.
- AUC: 1.0 = savršeno, 0.5 = nasumično, < 0.5 = gore od slučajnog modela.

Precision–Recall kriva

- Prikladnija je od ROC krive kada je pozitivna klasa retka, npr. alarm = 1% podataka.
- AP (Average Precision) je površina ispod P–R krive i bolji je pokazatelj za neuravnotežene skupove.
- U bezbednosti je prioritet visok recall — ne propustiti alarme — uz prihvatljivu preciznost.

L8-3: ROC kriva i Precision–Recall kriva za bezbednosne sisteme

Co-funded by the European Union

Python — ROC i PR krive za poređenje modela

```
from sklearn.metrics import RocCurveDisplay, PrecisionRecallDisplay
import matplotlib.pyplot as plt
fig, axes = plt.subplots(1, 2, figsize=(14, 5))
for naziv, model in finalni_modeli.items():
    y_prob = model.predict_proba(X_test)[:,1]
    RocCurveDisplay.from_predictions(y_test, y_prob,
        name=naziv, ax=axes[0])
    PrecisionRecallDisplay.from_predictions(y_test, y_prob,
        name=naziv, ax=axes[1])
axes[0].set_title('ROC krive'); axes[1].set_title('Precision-Recall krive')
plt.tight_layout(); plt.savefig('model_poredjenje.pdf', dpi=150)
```

L8-3: Maple — simbolička analiza klasifikacionih granica

Co-funded by the European Union

Maple — vizuelizacija granica odlučivanja (2D slučaj)

```
restart:
with(plots): with(Statistics):
# Generisi 2-klasne sinteticke podatke:
n := 100:
X1 := [Sample(Normal(1,1), n)[], Sample(Normal(-1,1), n)[]]:
X2 := [Sample(Normal(1,1), n)[], Sample(Normal(-1,1), n)[]]:
Y  := [seq(1,i=1..n), seq(0,i=1..n)]:
# Logisticka regresija simbolicki:
w1 := 1.5: w2 := 1.2: b := 0.1:
sigma := z -> 1/(1+exp(-z)):
P := (x1,x2) -> sigma(w1*x1 + w2*x2 + b):
# Granica odlucivanja: P(x1,x2) = 0.5 <=> w1*x1 + w2*x2 + b = 0:
x2_granica := x1 -> -(w1*x1 + b)/w2:
p_granica := plot(x2_granica(x1), x1=-4..4, color=red, thickness=3,
                  legend='Granica odlucivanja'):
# Konturni plot verovatnoce:
p_kontura := contourplot(P(x1,x2), x1=-4..4, x2=-4..4,
                         contours=[0.1,0.3,0.5,0.7,0.9],
                         colorscheme=[zgradient,['blue','white','red']]):
display([p_kontura, p_granica], title='Logisticka regresija – granica odlucivanja'):
```

L8-3: Objašnjivost modela — SHAP i LIME za procesnu bezbednost

Co-funded by the European Union

Zašto je objašnjivost ključna u bezbednosnim sistemima?

- Operater mora znati zašto model signalizira alarm — ne može slepo verovati „crnoj kutiji”.
- EU AI Act (2024): AI sistemi u bezbednosti moraju biti objašnjivi — to postaje regulatorni zahtev.
- Inženjeri moraju biti u stanju da otklone greške u modelu i razumeju njegove pogrešne odluke.

SHAP

- SHAP je zasnovan na Shapleyjevim vrednostima iz teorije igara — svaka karakteristika dobija „kredit” za predikciju.
- Globalna važnost karakteristika: zbir apsolutnih SHAP vrednosti kroz sve primere.
- Lokalna objašnjivost: za jedan konkretan alarm — koje karakteristike su doprinele i koliko.

L8-3: Objašnjivost modela — SHAP i LIME za procesnu bezbednost

Co-funded by the European Union

Python — SHAP waterfall plot za jedan alarm

```
import shap; import matplotlib.pyplot as plt
from xgboost import XGBClassifier
xgb = XGBClassifier(n_estimators=100).fit(X_train, y_train)
explainer = shap.TreeExplainer(xgb)
shap_values = explainer(X_test)
# Globalni summary plot:
shap.summary_plot(shap_values.values[:,1], X_test,
                  feature_names=feature_names, plot_type='bar')
# Lokalni waterfall za jedan alarm (primer 42):
shap.waterfall_plot(shap_values[42,:])
# Force plot – interaktivni HTML:
shap.force_plot(explainer.expected_value[1], shap_values.values[42,:,1],
                X_test.iloc[42], feature_names=feature_names)
```

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-4 Nadzirano učenje — regresija i odabir modela

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-4: Nadzirano učenje — regresija za procesne parametre

Co-funded by the European Union

Regresioni problem u procesnoj bezbednosti

- Cilj: predvideti kontinualnu promenljivu — temperaturu, pritisak, koncentraciju ili protok.
- Primene: predviđanje temperature reaktora 10 min unapred, prognoza nivoa emisija i modelovanje disperzije.
- ML surogatni modeli za CFD: zamena skupih CFD simulacija (L6) brzim ML modelima.

Linearna regresija — osnova

- $\hat{y} = w_0 + w_1x_1 + \ldots + w_nx_n = w^Tx + b$.
- Minimizuje MSE: $(1/m)\ \Sigma(y_i - \hat{y}_i)^2$ — OLS (Ordinary Least Squares) rešenje.
- Analitičko rešenje: $w^* = (X^TX)^{-1}X^Ty$ — jedinstven minimum.
- Ridge regresija (L_2): min MSE + $\lambda||w||_2^2$ — prevencija preprilagođavanja.
- Lasso regresija (L_1): min MSE + $\lambda||w||_1$ — smanjuje neke težine tačno na 0 (odabir karakteristika).

L8-4: Nadzirano učenje — regresija za procesne parametre

Co-funded by the European Union

Python — predviđanje temperature reaktora

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score
ridge = Ridge(alpha=10.0)
ridge.fit(X_train, y_train)
y_pred = ridge.predict(X_test)
print(f'RMSE: {mean_squared_error(y_test,y_pred)**0.5:.3f}')
print(f'R2: {r2_score(y_test,y_pred):.4f}')
```

L8-4: Nelinearna regresija — Gaussian Process i Random Forest regresija

Co-funded by the European Union

Gaussian Process Regression (GPR)

- Bajesovski pristup: model je raspodela nad funkcijama, a ne samo jedna funkcija.
- Daje predikciju i interval pouzdanosti — ključno za industrijsku bezbednost.
- Kernel funkcija određuje apriorne pretpostavke o „obliku" funkcije (RBF, Matérn, Periodic).
- Skalira kao $O(n^3)$ sa brojem uzoraka — nije pogodan za velike skupove (> 10000 uzoraka).

Random Forest Regressor

- Isti princip kao kod klasifikacije — ali listovi stabala predviđaju srednju vrednost umesto klase.
- Praktično: odličan za nelinearne zavisnosti u industrijskim podacima.
- Važnost karakteristika je dostupna — možemo videti koji senzori najviše utiču na izlaz.

L8-4: Nelinearna regresija — Gaussian Process i Random Forest regresija

Co-funded by the European Union

Python — GPR sa intervalima predviđanja i RF za predikciju koncentracije CO

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, WhiteKernel
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt; import numpy as np
kernel = 1.0*RBF(length_scale=1.0) + WhiteKernel(noise_level=0.1)
gpr = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=5)
gpr.fit(X_train, y_train)
y_pred, y_std = gpr.predict(X_test, return_std=True)
# 95% CI:
plt.fill_between(range(len(y_test)), y_pred-2*y_std, y_pred+2*y_std, alpha=0.3)
plt.plot(y_test, 'r.', label='Stvarna vrednost')
plt.plot(y_pred, 'b-', label='GPR predikcija')
```

Co-funded by the European Union

Surogatni (zamenski) modeli za SEVESO proračune

- Gaussov disperzioni model (L6-3) traži oko 0.1 s da izračuna koncentraciju u jednoj tački.
- QRA (L6-6) zahteva milione poziva modela — 0.1 s × 10^6 = 100000 s = 28 h, što je previše.
- Rešenje: naučiti ML model koji aproksimira C(x, y, Q, u, klasa) — jednom istreniran daje trenutnu predikciju.

Generisanje trening podataka iz fizičkog modela (Latin Hypercube Sampling)

```
from scipy.stats import qmc; import numpy as np
# LHS uzorkovanje ulaznih parametara:
sampler = qmc.LatinHypercube(d=5, seed=42)
sample = sampler.random(n=5000)
lower = [0.01, 1, 0, 1, 0]; upper = [5, 100, 30, 10, 5]
# Skale: Q[kg/s], x[m], H[m], u[m/s], klasa_PG[0-5]
params = qmc.scale(sample, lower, upper)
# Proracun C za svaku kombinaciju parametara:
C_labels = np.array([gauss_osa(p[1], p[0]*1e6, p[2], p[3], 'ABCDEF'[int(p[4])])
for p in params])
```

Co-funded by the European Union

Treniranje surogatnog modela

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
surrogate = GradientBoostingRegressor(n_estimators=500, max_depth=5)
surrogate.fit(params, np.log1p(C_labels))  # log-transformacija za bolji fit
cv_r2 = cross_val_score(surrogate, params, np.log1p(C_labels), cv=5,
scoring='r2')
print(f'Surrogate R2 = {cv_r2.mean():.4f} — brzina: ~1000x brže od fizičkog
modela!')
```

L8-4: Bajesovska optimizacija hiperparametara

Co-funded by the European Union

Zašto je Grid Search loš?

- Grid Search: iscrpno pretraživanje mreže hiperparametara — eksponencijalno skupo.
- 10 hiperparametara × 5 vrednosti = 5^{10} = 9.7 miliona kombinacija — praktično nemoguće.
- Random Search: nasumično uzorkovanje — bolje, ali i dalje neefikasno jer ne koristi prethodne rezultate.

Bajesovska optimizacija

- Bajesovska optimizacija gradi probabilistički model (obično GP) funkcije cilja.
- Biramo sledeći skup hiperparametara koji balansira exploration (neistraženo) i exploitation (već dobro).
- Optuna, Hyperopt i Scikit-Optimize: biblioteke za Bajesovsku optimizaciju u Pythonu.

Co-funded by the European Union

Python — Optuna za podešavanje XGBoost modela na procesnim podacima

```
import optuna; from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
optuna.logging.set_verbosity(optuna.logging.WARNING)
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 500),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_float('lr', 0.01, 0.3, log=True),
        'subsample': trial.suggest_float('sub', 0.5, 1.0),
    }
    m = XGBClassifier(**params, use_label_encoder=False, eval_metric='logloss')
    return cross_val_score(m, X_train, y_train, cv=3, scoring='f1').mean()
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
print('Najbolji F1:', study.best_value, '| Parametri:', study.best_params)
```

L8-4: MAPLE — analiza osetljivosti ML modela

Co-funded by the European Union

MAPLE — simbolička analiza osetljivosti surogatnog modela

```
restart:
# Pretpostavimo fitovani polinom 2. reda za log(C) = f(Q, x, u):
# C = exp(a0 + a1*Q + a2*x + a3*u + a4*Q*x + a5*x*u + ...)
a := [3.2, 0.8, -1.5, -0.4, -0.2, 0.1, -0.05]:
logC := (Q, x, u) -> a[1] + a[2]*Q + a[3]*log(x) + a[4]*u
          + a[5]*Q*log(x) + a[6]*log(x)*u + a[7]*Q*u:
C := (Q, x, u) -> exp(logC(Q, x, u)):

# Sobolova analiza osetljivosti:
# Parcijalni izvodi za svaki parametar:
dC_dQ := diff(C(Q, x, u), Q):
dC_dx := diff(C(Q, x, u), x):
dC_du := diff(C(Q, x, u), u):
# Evaluacija u tipičnoj tački:
Q0:=1: x0:=500: u0:=4:
printf('dC/dQ = %.4f | dC/dx = %.4f | dC/du = %.4f\n',
       eval(subs([Q=Q0,x=x0,u=u0], dC_dQ)),
       eval(subs([Q=Q0,x=x0,u=u0], dC_dx)),
       eval(subs([Q=Q0,x=x0,u=u0], dC_du))):
# Zaključak: koji parametar modela izvora najviše utiče na zonu hazarda?
```

Co-funded by the European Union

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-5 Neuronske mreže i duboko učenje

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-5: Veštačke neuronske mreže — osnove

Co-funded by the European Union

Inspiracija iz neurologije — veoma gruba analogija

- Neuron prima ulaze, obrađuje ih i šalje izlaz ako je aktiviran.
- Veštački neuron: $y = f(w_1x_1 + \ldots + w_nx_n + b) = f(w^Tx + b)$.

Aktivacione funkcije

- Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$ — opseg (0, 1); problem je nestajući gradijent u dubokim mrežama.
- Tanh: $\tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$ — opseg (−1, 1), centriran oko 0.
- ReLU: max(0, z) — najčešći izbor za skrivene slojeve, brz i stabilan.
- Leaky ReLU: max(0.01z, z) — ublažava problem „mrtvih neurona”.
- Softmax: poslednji sloj višeklasnih klasifikatora; zbir verovatnoća je 1.0.

Višeslojni perceptron (MLP — Multi-Layer Perceptron)

- Ulazni sloj → skriveni slojevi → izlazni sloj; dubina je broj slojeva.
- Svaki sloj: linearna transformacija + aktivaciona funkcija.
- Teorema univerzalne aproksimacije: MLP sa jednim dovoljno širokim skrivenim slojem može aproksimirati bilo koju kontinualnu funkciju.
- U praksi su često bolji dublji modeli sa više užih slojeva.

Co-funded by the European Union

Algoritam učenja — Forward Pass

- Ulaz x prolazi kroz sve slojeve unapred: $x \rightarrow L_1 \rightarrow L_2 \rightarrow \ldots \rightarrow \hat{y}$.
- Svaki sloj: $z = W \cdot a_{prev} + b$, $a = f(z)$, gde je f aktivaciona funkcija.
- Greška: $L = loss(y, \hat{y})$, npr. cross-entropy za klasifikaciju ili MSE za regresiju.

Backward Pass (Backpropagation)

- Lančano pravilo: $\partial L/\partial w = (\partial L/\partial y) \cdot (\partial y/\partial z) \cdot (\partial z/\partial w)$.
- Gradijenti se propagiraju od izlaza ka ulazu — otuda naziv backward.
- Ažuriranje težina: $w = w - \alpha \cdot \partial L/\partial w$ — jedan korak gradijentnog spusta.

Co-funded by the European Union

Optimizatori

- SGD: korak na jednom mini-batchu — brz, ali bučan.
- Adam: adaptivna stopa učenja za svaki parametar — često preporučen početni izbor.
- Learning-rate schedule: početi većom stopom učenja i smanjivati je tokom treninga.

Python — MLP za klasifikaciju kvarova reaktora

```
import tensorflow as tf; from tensorflow import keras
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(4, activation='softmax')  # 4 klase kvarova
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
hist = model.fit(X_train, y_train, epochs=100, batch_size=64,
                 validation_split=0.2,
callbacks=[keras.callbacks.EarlyStopping(patience=10)])
```

L8-5: Regularizacija neuronskih mreža

Co-funded by the European Union

Zašto regularizacija neuronskih mreža?

- Mreže sa milionima parametara lako memorišu trening podatke umesto da nauče opšte obrasce.
- Regularizacija smanjuje efektivnu kompleksnost modela i poboljšava generalizaciju.

Dropout

- Nasumično isključujemo p% neurona tokom treninga u svakom koraku, tipično p = 0.2–0.5.
- Efekat: trenira se veliki broj različitih podmreža, što smanjuje korelaciju neurona.
- Tokom testiranja svi neuroni su uključeni, a težine se skaliraju sa (1 − p).

Batch Normalization

- Normalizuje aktivacije svakog sloja na srednju vrednost 0 i standardnu devijaciju 1 tokom mini-batcha.
- Prednosti: brže treniranje, manja osetljivost na inicijalizaciju i blaga regularizacija.
- Standardno se koristi u modernim neuronskim mrežama i često poboljšava performanse.

Co-funded by the European Union

Early Stopping

- Prati se validacioni gubitak i trening se zaustavlja kada gubitak počne da raste.

Python — kompletan trening sa regularizacijom

```
callbacks = [
    keras.callbacks.EarlyStopping(monitor='val_loss', patience=15,
restore_best_weights=True),
    keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=5),
    keras.callbacks.ModelCheckpoint('best_model.h5', save_best_only=True)
]
hist = model.fit(X_tr, y_tr, epochs=200, batch_size=32,
                 validation_data=(X_val, y_val), callbacks=callbacks, verbose=0)
# Kriva ucenja:
plt.plot(hist.history['loss'], label='Train loss')
plt.plot(hist.history['val_loss'], label='Val loss')
plt.legend(); plt.xlabel('Epoha'); plt.ylabel('Loss')
```

L8-5: Konvolutivne neuronske mreže (CNN) za vizuelnu inspekciju

Co-funded by the European Union

CNN za procesnu bezbednost — primena na slike

- Detekcija curenja iz video-snimka: kamera na postrojenju, CNN klasifikuje kadrove.
- Detekcija korozije i oštećenja cevi pomoću drona ili inspekcijskog robota.
- Automatsko očitavanje analognih instrumenata, npr. pritiskomera i termometara, pomoću OCR + CNN.

CNN arhitektura

- Konvolutivni slojevi uče lokalne obrasce: ivice, teksture i oblike pomoću malih filtera (3×3, 5×5).
- Pooling slojevi (MaxPool): smanjenje dimenzionalnosti i translaciona invarijantnost.
- Flatten + Dense slojevi: klasifikacija na osnovu naučenih karakteristika.
- Transfer learning: koristiti prethodno treniran CNN (ResNet, EfficientNet) za industrijske slike.

L8-5: Konvolutivne neuronske mreže (CNN) za vizuelnu inspekciju

Co-funded by the European Union

Python — transfer learning za detekciju curenja na slikama

```
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers, Model
base = EfficientNetB0(include_top=False, weights='imagenet',
input_shape=(224,224,3))
base.trainable = False  # zamrzni pretrenisane tezine
x = base.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.3)(x)
out = layers.Dense(2, activation='softmax')(x)  # curenje / bez curenja
model = Model(inputs=base.input, outputs=out)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

L8-5: Autoenkoder za detekciju anomalija bez oznaka

Co-funded by the European Union

Autoenkoder — nenadzirano učenje anomalija

- Arhitektura: enkoder X → z kompresuje podatke u latentni prostor; dekoder z → X̂ rekonstruiše ulaz.
- Trenira se samo na normalnim podacima — uči efikasno predstavljanje normalnog ponašanja.
- Na testu: rekonstrukciona greška = $||X - \hat{X}||^2$; visoka greška označava anomaliju.
- Ključna prednost: nisu potrebni označeni podaci o kvarovima, koji često nedostaju u industriji.

Varijanta: Variational Autoencoder (VAE)

- Latentni prostor je raspodela (μ, σ), a ne jedna tačka — probabilistički pristup.
- VAE može generisati sintetičke normalne uzorke za augmentaciju skupa podataka.

L8-5: Autoenkoder za detekciju anomalija bez oznaka

Co-funded by the European Union

Python — Autoencoder za detekciju anomalija u SCADA podacima

```
from tensorflow import keras; import numpy as np
n_features = X_normal.shape[1]
# Enkoder:
inp = keras.Input(shape=(n_features,))
enc = keras.layers.Dense(64, activation='relu')(inp)
enc = keras.layers.Dense(32, activation='relu')(enc)
lat = keras.layers.Dense(8, activation='relu')(enc)  # latentni prostor
# Dekoder:
dec = keras.layers.Dense(32, activation='relu')(lat)
dec = keras.layers.Dense(64, activation='relu')(dec)
out = keras.layers.Dense(n_features)(dec)
ae = keras.Model(inp, out)
ae.compile(optimizer='adam', loss='mse')
ae.fit(X_normal, X_normal, epochs=100, batch_size=64, validation_split=0.1)
rec_err = np.mean((X_test - ae.predict(X_test))**2, axis=1)
prag = np.percentile(rec_err[:len(X_normal)], 99)  # 99. percentil normale
y_pred = (rec_err > prag).astype(int)  # 1 = anomalija
```

L8-5: Fizički informisane neuronske mreže (PINN)

Co-funded by the European Union

Šta su PINN?

- PINN (Physics-Informed Neural Networks) su neuronske mreže koje u loss funkciji poštuju fizičke zakone.
- Razvijeno i popularizovano radovima Raissija i saradnika — snažan pristup za naučne probleme.
- Ključna ideja: kombinovati fleksibilnost neuronskih mreža sa fizičkom konzistentnošću.
- Loss = L_data + λ·L_fizika, gde L_fizika meri koliko model krši fizički zakon.

Primena PINN u procesnoj bezbednosti

- Predviđanje disperzije gasa: PINN uči iz merenja, ali mora poštovati difuzionu jednačinu.
- Prenos toplote u reaktoru: model poštuje Fourierov zakon i ostaje fizički konzistentan.
- Termičko bekstvo: PINN uči kinetiku reakcije iz malog broja eksperimentalnih podataka.

Co-funded by the European Union

Python — PINN za 1D difuzionu jednačinu

```
import tensorflow as tf; import numpy as np
# c_t = D * c_xx  (difuziona jednacina, D=konstanta)
D = 0.01  # difuzioni koeficijent
# Gradijenti se racunaju sa GradientTape:
@tf.function
def pinn_loss(model, x_t_col, x_t_data, c_data):
    x, t = x_t_col[:,0:1], x_t_col[:,1:2]
    with tf.GradientTape(persistent=True) as tape:
        tape.watch([x, t])
        c = model(tf.concat([x,t], axis=1))
        c_x = tape.gradient(c, x)
    c_xx = tape.gradient(c_x, x)
    c_t  = tape.gradient(c, t)
    L_phy = tf.reduce_mean((c_t - D*c_xx)**2)
    L_dat = tf.reduce_mean((model(x_t_data) - c_data)**2)
    return L_dat + 0.1*L_phy
```

L8-5: Studija slučaja — NN za predikciju temperature kliznih ležaja

Co-funded by the European Union

Scenario

- Predviđanje temperature ležaja 30 min unapred na centrifugalnoj pumpi.
- Podaci: $T_{ležaj}$, T_{ulje}, $T_{ambijent}$, vibracije X/Y, brzina i protok — satna merenja tokom 2 godine.
- Kvar nastaje kada $T_{ležaj}$ pređe 95 °C; cilj je upozorenje 30 min pre toga.

Priprema podataka — lagged features, odnosno vremenski prozori

```
import pandas as pd; import numpy as np
def napravi_prozore(df, ulazni_koraci=30, izlazni_koraci=6):
    X, y = [], []
    for i in range(len(df) - ulazni_koraci - izlazni_koraci):
        X.append(df.iloc[i:i+ulazni_koraci].values)  # prozor od 30 uzoraka
        y.append(df['T_lezaj'].iloc[i+ulazni_koraci+izlazni_koraci])  # ciljni
uzorak
    return np.array(X), np.array(y)
X_win, y_win = napravi_prozore(df_sensor, 30, 6)
# X_win.shape = (n_uzoraka, 30, n_features)
```

L8-5: Studija slučaja — NN za predikciju temperature kliznih ležaja

Co-funded by the European Union

Scenario

- Predviđanje temperature ležaja 30 min unapred na centrifugalnoj pumpi.
- Podaci: T_ležaj, T_ulje, T_ambijent, vibracije X/Y, brzina i protok — satna merenja tokom 2 godine.
- Kvar nastaje kada T_ležaj pređe 95 °C; cilj je upozorenje 30 min pre toga.

Priprema podataka — lagged features, odnosno vremenski prozori

```
import pandas as pd; import numpy as np
def napravi_prozore(df, ulazni_koraci=30, izlazni_koraci=6):
    X, y = [], []
    for i in range(len(df) - ulazni_koraci - izlazni_koraci):
        X.append(df.iloc[i:i+ulazni_koraci].values)  # prozor od 30 uzoraka
        y.append(df['T_lezaj'].iloc[i+ulazni_koraci+izlazni_koraci])  # ciljni
uzorak
    return np.array(X), np.array(y)
X_win, y_win = napravi_prozore(df_sensor, 30, 6)
# X_win.shape = (n_uzoraka, 30, n_features)
```

L8-5: Studija slučaja — NN za predikciju temperature kliznih ležaja

Co-funded by the European Union

MLP sa uravnatim prozorima

```
from tensorflow import keras
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(30, n_features)),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.BatchNormalization(), keras.layers.Dropout(0.3),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(1)  # regresija – predikcija T
])
model.compile(optimizer='adam', loss='huber')  # Huber loss: robustna na
autlaire
```

L8-5: MAPLE — simbolička analiza NN aktivacione funkcije i gradijenta

Co-funded by the European Union

MAPLE — analiza aktivacionih funkcija i derivata

```
restart:
# Definisanje aktivacionih funkcija:
relu  := x -> piecewise(x >= 0, x, 0):
lrelu := x -> piecewise(x >= 0, x, 0.01*x):
sig   := x -> 1/(1+exp(-x)):
tanh_f := x -> tanh(x):

# Derivati (kljucni za backpropagation):
d_sig   := diff(sig(x), x):
d_tanh  := diff(tanh_f(x), x):
simplify(d_sig);   # sig(x)*(1-sig(x))
simplify(d_tanh);  # 1 - tanh(x)^2
```

L8-5: MAPLE — simbolička analiza NN aktivacione funkcije i gradijenta

Co-funded by the European Union

MAPLE — analiza aktivacionih funkcija i derivate…..NASTAVAK

```
# Problem vanishing gradient za duboke sigmoid mreze:
# Maksimalni gradijent sig: d_sig(0) = 0.25 -> posle 10 slojeva: 0.25^10 = 9.5e-7 !
printf('Max gradient sig posle 10 slojeva: %.2e\n', 0.25^10):
printf('Max gradient relu posle 10 slojeva: %.2e\n', 1.0^10):

with(plots):
p1 := plot(sig(x), x=-5..5, color=blue, legend='Sigmoid'):
p2 := plot(tanh_f(x), x=-5..5, color=red, legend='Tanh'):
p3 := plot(relu(x), x=-5..5, color=green, legend='ReLU'):
display([p1,p2,p3], title='Aktivacione funkcije – poredjenje'):
```

Co-funded by the European Union

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-6 Nenadzirano učenje — klasterovanje i detekcija anomalija

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

L8-6: Nenadzirano učenje — klasterovanje procesnih stanja

Co-funded by the European Union

K-Means klasterovanje

- Algoritam: izaberi k centara nasumično | Dodeli svaki primer najbližom centru | Ažuriraj centar
- Ponavljaj dok se centri ne stabilizuju (konvergencija) | Konvergira za O(n · k · iter) korak
- Primena: Grupisanje sličnih operativnih stanja reaktora — bez unapred poznatih labela!

Odabir k — Elbow metoda

- Iscrtati Within-Cluster Sum of Squares (WCSS) vs k | Lakat krive = optimalni k
- Silhouette score: mera kvaliteta klastera u [-1, 1] — što visi to bolji

L8-6: Nenadzirano učenje — klasterovanje procesnih stanja

Co-funded by the European Union

K-Means klasterovanje

- Algoritam: izaberi k centara nasumično | Dodeli svaki primer najbližom centru | Ažuriraj centar
- Ponavljaj dok se centri ne stabilizuju (konvergencija) | Konvergira za O(n · k · iter) korak
- Primena: Grupisanje sličnih operativnih stanja reaktora — bez unapred poznatih labela!

Odabir k — Elbow metoda:

- Iscrtati Within-Cluster Sum of Squares (WCSS) vs k | Lakat krive = optimalni k
- Silhouette score: mera kvaliteta klastera u [-1, 1] — što viši to bolji

Python — K-Means za grupisanje operativnih stanja pumpe:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt; import numpy as np
X_scaled = StandardScaler().fit_transform(X_pump)
wcss = []; sil = []
for k in range(2, 11):
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(X_scaled)
    wcss.append(km.inertia_)
    sil.append(silhouette_score(X_scaled, labels))
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(12,4))
ax1.plot(range(2,11), wcss, 'bo-'); ax1.set_title('Elbow metoda')
ax2.plot(range(2,11), sil, 'rs-'); ax2.set_title('Silhouette score')
```

L8-6: DBSCAN i detekcija anomalija nenadziranim pristupom

Co-funded by the European Union

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- Za razliku od K-Means: ne treba unapred zadati k, pronalazi klastere proizvoljnog oblika
- Parametri: eps (radijus susedstva), $min_{samples}$ (minimalan broj tačaka u susedstvu)
- Tačke van gustih regiona su označene kao NOISE = -1 — ovo su anomalije!
- DBSCAN je direktno upotrebljiv za detekciju anomalija u procesnim podacima

Isolation Forest — najbrža metoda za detekciju anomalija

- Ideja: anomalije je lako 'izolovati' od ostatka podataka (trebaju mali broj pitanja)
- Slučajno gradi stabla i broji korake do izolacije — mali broj koraka = anomalija
- Contamination parametar: procenjeni udeo anomalija u podacima

Python — Isolation Forest i DBSCAN za detekciju anomalija u SCADA podacima:

```
from sklearn.ensemble import IsolationForest
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import numpy as np; import matplotlib.pyplot as plt
X_s = StandardScaler().fit_transform(X_scada)
# Isolation Forest:
iforest = IsolationForest(n_estimators=200, contamination=0.02, random_state=42)
iforest.fit(X_normal)  # treniraj SAMO na normalnim podacima
y_if = iforest.predict(X_s)  # -1 = anomalija, 1 = normalno
print(f'IF anomalije: {(y_if==-1).sum()} od {len(y_if)}')
# DBSCAN:
db = DBSCAN(eps=0.5, min_samples=10)
labels = db.fit_predict(X_s)
print(f'DBSCAN noise tacke: {(labels==-1).sum()}')
```

L8-6: Local Outlier Factor i One-Class SVM

Co-funded by the European Union

LOF (Local Outlier Factor)

- Meri lokalnu gustinu svake tačke u odnosu na njene susede
- LOF ≫ 1: tačka je u regiji znatno manje gustine od njenih suseda = anomalija
- Osetljiv na $k_{neighbors}$ parametar — kros-validovati ili koristiti opseg k = [10, 20, 30]

One-Class SVM

- Uči granicu oko normalnih podataka — sve izvan granice je anomalija
- Kernel trik (RBF) za nelinearne granice | Nu: gornja granica udela outlaiera
- Sporiji od Isolation Forest za velike skupove, ali tačan za niske dimenzionalnosti

Poređenje metoda detekcije anomalija

- Isolation Forest: brz, skalira za velike skupove, manje parametara — PREPORUČEN za SCADA
- LOF: dobar za lokalne anomalije — kvar jednog senzora dok ostali su normalni
- DBSCAN: dobro grupiše ali osetljiv na eps i $min_{samples}$
- Autoencoder: najtačniji za visoko-korelisane podatke, ali zahteva treniranje NN

Python — poredjenje svih metoda:

```
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
metode = {
    'IF':  IsolationForest(contamination=0.02, n_estimators=200),
    'LOF': LocalOutlierFactor(n_neighbors=20, contamination=0.02),
    'OC-SVM': OneClassSVM(kernel='rbf', nu=0.02)
}
for naziv, metoda in metode.items():
    y_pred = metoda.fit_predict(X_normal)  # -1=anomalija za sve
    # Evaluacija na poznatom test skupu:
    if hasattr(metoda, 'decision_function'):
        score = roc_auc_score(y_true_test, metoda.decision_function(X_test))
        print(f'{naziv}: AUC = {score:.3f}')
```

L8-6: Alarm Management sa ML — redukcija lažnih alarma

Co-funded by the European Union
Safety 4EU

Problem lažnih alarma u procesnim postrojenjima

- Studija: prosečno postrojenje ima > 80% lažnih alarma — operater razvija 'alarm blindness'
- ISA-18.2 standard: maksimum 6 alarma/operater/sat u stabilnom radu
- ML pristup: naučiti model da razlikuje prave od lažnih alarma na osnovu konteksta

Karakteristična obeležja za alarm klasifikaciju

- Vrednost alarma, smer trenda, brzina promene, istorija poslednjeg alarma, korelisani alarmi
- Vreme od poslednjeg alarma na istom tagu — frekventni alarm = verovatno lažni
- Kontekstualna stanja postrojenja: startovanje, zatvaranje, plansko održavanje

Python — ML za klasifikaciju alarma:

```
import pandas as pd; from sklearn.ensemble import RandomForestClassifier
# Ucitaj alarm log:
alarms = pd.read_csv('alarm_log.csv', parse_dates=['timestamp'])
# Feature engineering za svaki alarm:
alarms['time_since_last'] = alarms.groupby('tag')['timestamp'].diff().dt.total_seconds()
alarms['alarm_rate_1h'] = alarms.groupby('tag')['timestamp'].transform(
    lambda x: x.rolling('1h').count())
alarms['trend'] = alarms.groupby('tag')['value'].transform(lambda x: x.diff())
features = ['value','time_since_last','alarm_rate_1h','trend','duration_s']
X = alarms[features].fillna(0); y = alarms['is_nuisance']
clf = RandomForestClassifier(n_estimators=200, class_weight='balanced')
scores = cross_val_score(clf, X, y, cv=5, scoring='f1')
print(f'F1 za lazni alarm: {scores.mean():.3f}')
```

L8-6: MAPLE — klasterovanje i vizuelizacija operativnih stanja

Co-funded by the European Union

MAPLE — K-Means analiza procesnih stanja

```
restart:
with(Statistics): with(plots): with(LinearAlgebra):
# Simulovani podaci 3 operativna stanja pumpe:
# Stanje 0: normalan rad, Stanje 1: kavitacija, Stanje 2: kvar lezaja
T_norm  := Sample(Normal(60, 3), 300): V_norm  := Sample(Normal(2, 0.3), 300):
T_kav   := Sample(Normal(70, 5), 100): V_kav   := Sample(Normal(2.8, 0.5), 100):
T_lezaj := Sample(Normal(85, 8), 80):  V_lezaj := Sample(Normal(3.5, 0.8), 80):
T_all := [op(T_norm), op(T_kav), op(T_lezaj)]:
V_all := [op(V_norm), op(V_kav), op(V_lezaj)]:
# Iscrtati sve podatke:
p_podaci := pointplot(zip(T_norm, V_norm), color=blue,  symbol=point, symbolsize=5):
p_kav    := pointplot(zip(T_kav,  V_kav),  color=orange,symbol=point, symbolsize=5):
p_lezaj  := pointplot(zip(T_lezaj,V_lezaj),color=red,   symbol=point, symbolsize=5):
display([p_podaci, p_kav, p_lezaj],
        labels=['T_lezaj [degC]','Vibracija [mm/s]'],
        title='Operativna stanja pumpe u prostoru features'):
```

Safety4EU projekat
L8. Uvod u mašinsko učenje na radnom mestu i bezbednost procesa

L8-7 Vremenske serije i LSTM za predviđanje hazarda

Dr Milan Gocić, Dr Milan Protić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Co-funded by the European Union

Karakteristike industrijskih vremenskih serija

- Sezonalnost: dnevni/nedeljni/godišnji ciklusi u energetici, hemijskim postrojenjima
- Trend: postepeno degradacija opreme — T ležaja polako raste tokom meseci
- Autokorelacija: vrednost u t je slična vrednosti u t-1, t-2 itd. — ne nezavisni uzorci!
- Stacionarnost: statističke karakteristike ne menjaju se tokom vremena — zahtev za ARIMA

Klasični modeli vremenskih serija

- AR(p) (Auto-Regressive): $y_t = c + phi_1 \cdot y_{t-1} + \ldots + phi_p \cdot y_{t-p} + epsilon_t$
- MA(q) (Moving Average): $y_t = \mu + epsilon_t + theta_1 \cdot epsilon_{t-1} + \ldots$
- ARIMA(p,d,q): AR + diferenciranje (za stacionarnost) + MA — klasičan statistički model
- Ograničenja klasičnih modela: linearnost, stacionarnost, ručno podešavanje p,d,q

Co-funded by the European Union

Python — dekompozicija i ARIMA:

```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
# Dekompozicija na trend + sezonalnost + rezidual:
res = seasonal_decompose(df['T_reaktor'], model='additive', period=24)
res.plot(); plt.tight_layout()
# ARIMA za prognozu:
arima = ARIMA(df['T_reaktor'], order=(3,1,2))
fit = arima.fit()
forecast = fit.forecast(steps=48)  # prognoza 48h unapred
```

L8-7: LSTM mreže — pamćenje dugih sekvenci

Co-funded by the European Union

Problem vanišujućeg gradijenta u RNN

- Rekurentne mreže (RNN) rade na sekvencama, ali imaju problem: gradijent se gasi unazad!
- Posle ~10-20 vremenskih koraka, mreža 'zaboravlja' ranije informacije
- Problem: industrijske anomalije mogu biti najavljene događajima od pre sati!

LSTM (Long Short-Term Memory) rešenje (Hochreiter & Schmidhuber, 1997)

- LSTM čvor ima 3 kapije (gates) koji kontrolišu pamćenje i zaboravljanje:
- Kapija zaboravljanja (Forget gate): $ft = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ — šta zaboraviti?
- Kapija ulaza (Input gate): $it = \sigma(W_i*[h_{t-1},xt]+bi)$ — šta novo pamtiti?
- Kapija izlaza (Output gate): $ot = \sigma(W_o*[h_{t-1},xt]+bo)$ — šta izlaziti?
- Stanje ćelije: $C_t = ft*C_{t-1} + it*\tanh(W_c*[h_{t-1},xt]+bc)$ — memorija

L8-7: LSTM mreže — pamćenje dugih sekvenci

Co-funded by the European Union

Python — LSTM za predviđanje T reaktora 1h unapred:

```
from tensorflow import keras; import numpy as np
model = keras.Sequential([
    keras.layers.LSTM(128, return_sequences=True, input_shape=(seq_len,
  n_feat)),
    keras.layers.Dropout(0.2),
    keras.layers.LSTM(64, return_sequences=False),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)  # predikcija T za sledeci korak
])
model.compile(optimizer='adam', loss='huber', metrics=['mae'])
hist = model.fit(X_train, y_train, epochs=50, batch_size=64,
                 validation_split=0.2, verbose=0)
```

L8-7: Transformer model za vremenske serije (napredna tema)

Co-funded by the European Union

Transformer arhitektura (Vaswani et al., 2017)

- Originalno razvijen za NLP (BERT, GPT) ali sada dominira i u vremenskim serijama
- Ključni mehanizam: Self-Attention — svaki vremenski korak 'pazi' na sve ostale korake
- Attention(Q,K,V) = softmax(Q*K^T / √(d_k)) * V — matematička forma
- Prednosti nad LSTM: paralelan trening, bolje za dugi rok zavisnosti, skalabilniji

Temporal Fusion Transformer (TFT)

- Razvijen od Google (Lim et al., 2020) specifično za multivarijantne vremenske serije
- Prednosti: interpretabilna pažnja, statički i dinamički features, kvantilna predikcija
- pytorch-forecasting biblioteka: TFT implementacija za industrijsku primenu

L8-7: Transformer model za vremenske serije (napredna tema)

Co-funded by the European Union

Python — jednostavan Transformer za vremensku seriju:

```
from tensorflow import keras
# Multi-head attention za vremenske serije:
inp = keras.Input(shape=(seq_len, n_feat))
x = keras.layers.MultiHeadAttention(num_heads=4, key_dim=32)(inp, inp)
x = keras.layers.LayerNormalization()(inp + x)  # residual + norm
x = keras.layers.GlobalAveragePooling1D()(x)
x = keras.layers.Dense(64, activation='relu')(x)
out = keras.layers.Dense(1)(x)
model = keras.Model(inp, out)
model.compile(optimizer='adam', loss='mse')
print(model.summary())
```

L8-7: Prediktivno održavanje — Remaining Useful Life (RUL) procena

Co-funded by the European Union

Remaining Useful Life (RUL) — preostali radni vek

- RUL(t) = T_{kvar} - t = vreme preostalo do kvara ako se nastavi sa trenutnim korišćenjem
- Cilj ML: predvideti RUL iz vremenskih serija senzorskih merenja
- Smanjena RUL = alarm za planiranje održavanja | RUL < 0 = kvar je nastao!

CMAPSS skup podataka (NASA) — benchmark za RUL procenu:

- Simulirani motori mlaznica sa kvarovima u različito vreme
- Slobodno dostupan: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

L8-7: Prediktivno održavanje — Remaining Useful Life (RUL) procena

Co-funded by the European Union

Python — RUL procena sa LSTM na CMAPSS podacima:

```
import pandas as pd; import numpy as np
# Ucitaj CMAPSS FD001:
col = ['unit','cycle','op1','op2','op3'] + [f's{i}' for i in
  range(1,22)]
train = pd.read_csv('train_FD001.txt', sep=' ', header=None, names=col)
# Izracunaj RUL za svaki vremenski korak:
max_cycle = train.groupby('unit')['cycle'].max().reset_index()
max_cycle.columns = ['unit', 'max_cycle']
train = train.merge(max_cycle, on='unit')
train['RUL'] = train['max_cycle'] - train['cycle']
# Ogranicit RUL na max 130 (nije korisno predvidjati vise od 130 ciklusa
  unapred):
train['RUL'] = train['RUL'].clip(upper=130)
print(train.groupby('unit')['RUL'].max().describe())
```

L8-7: Detekcija degradacije procesne opreme — rano upozoravanje

Co-funded by the European Union

Rano upozoravanje (Early Warning System — EWS)

- Cilj: detektovati početak degradacije ZNATNO pre nego sto dostiznemo kritičnu vrednost
- Trend analiza: Detecting Small Changes in Mean (CUSUM, EWMA kontrolne karte)
- ML pristup: naučiti tipičan 'potpis' degradacije iz istorijskih podataka kvarova

CUSUM (Cumulative Sum Control Chart)

- Akumulira sva mala odstupanja od srednje vrednosti — osetljiv na spore promene!
- $C+ = \max(0, C+_{prev} + (x - \mu - k))$ gde je k polu-dozvoljeno odstupanje
- Alarm: $C+ > H$ (granica alarma) | Tipično $H = 4\text{-}5\ \sigma$

L8-7: Detekcija degradacije procesne opreme — rano upozoravanje

Co-funded by the European Union

Python — CUSUM i EWMA rano upozoravanje:

```
import numpy as np; import matplotlib.pyplot as plt
def cusum(x, k=1.0, H=5.0):
    mu = np.mean(x[:100])  # prvih 100 uzoraka = referentno normalno
    sigma = np.std(x[:100])
    C_plus = np.zeros(len(x))
    for i in range(1, len(x)):
        C_plus[i] = max(0, C_plus[i-1] + (x[i]-mu)/sigma - k)
    return C_plus, C_plus > H  # signal i alarm flags
C, alarmi = cusum(df['T_lezaj'].values)
plt.subplot(2,1,1); plt.plot(df['T_lezaj']); plt.title('T lezaja')
plt.subplot(2,1,2); plt.plot(C, 'b-', label='CUSUM')
plt.axhline(5, color='r', ls='--', label='H=5')
plt.scatter(np.where(alarmi)[0], C[alarmi], c='r', zorder=5,
  label='Alarm')
plt.legend(); plt.tight_layout()
```

L8-7: Kompletna studija slučaja — ML sistem za praćenje kompresora

Co-funded by the European Union

Scenario: Sistem za rano upozoravanje na kvar kompresora u rafineriji

- Kompresori su 'srce' rafinerije — kvar može izazvati hitno zaustavljanje i SEVESO incident
- Senzori: $T_{le}z_{aj}$_A/B, P_{ula}z/izlaz, protok, brzina, $vibr_X$/Y/Z, struja motora (11 senzora)
- Podaci: 3 godine istorije, 3 planirana i 2 neplanirana kvara dokumentovana

Arhitektura ML sistema

- Sloj 1 (Detekcija): Isolation Forest na normalnim podacima -> anomaly score u realnom vremenu
- Sloj 2 (Klasifikacija): RF klasifikator -> tip anomalije (ležaj / labirint / ventil)
- Sloj 3 (Prognoza): LSTM -> RUL predikcija -> broj ciklusa do očekivanog kvara

L8-7: Kompletna studija slučaja — ML sistem za praćenje kompresora

Co-funded by the European Union

Python — integracija sva tri sloja

```
class KompresorMonitoring:
    def __init__(self, iforest, rf_clf, lstm_rul):
        self.iforest = iforest; self.clf = rf_clf; self.lstm = lstm_rul
    def analiziraj(self, novi_prozor):
        # Sloj 1: anomaly score
        score = -self.iforest.decision_function([novi_prozor[-1]])[0]
        if score > 0.3:  # potencijalna anomalija
            # Sloj 2: tip kvara
            features = extract_features(novi_prozor)
            tip = self.clf.predict([features])[0]
            # Sloj 3: RUL prognoza
            rul = self.lstm.predict(novi_prozor[np.newaxis])[0,0]
            return {'anomalija': True, 'tip': tip, 'RUL': rul, 'score':
 score}
        return {'anomalija': False}
```

L8-7: MAPLE — analiza vremenskih serija i autokorelaciona funkcija

Co-funded by the European Union

MAPLE — autokorelaciona funkcija i stacionarnost

```
restart:
with(Statistics): with(plots):
# Generisi AR(2) procesnu vremensku seriju:
# y_t = 0.7*y_{t-1} - 0.2*y_{t-2} + epsilon_t
n := 500: phi1 := 0.7: phi2 := -0.2:
eps := Sample(Normal(0, 1), n):
y := Array(1..n, 0.0):
for i from 3 to n do
    y[i] := phi1*y[i-1] + phi2*y[i-2] + eps[i]:
end do:
# Autokorelaciona funkcija (ACF):
ACF := k -> Statistics:-Correlation(y[1..n-k], y[1+k..n]):
lags := [seq(k, k=0..30)]:
acf_vals := map(ACF, lags):
p_acf := pointplot(zip(lags, acf_vals), symbol=solidbox, symbolsize=15,
                   color=blue, labels=['Lag','ACF']):
h_95 := plot([1.96/sqrt(n), -1.96/sqrt(n)], lag=0..30, color=red,
             linestyle=[dash,dash]):  # 95% CI granice
display([p_acf, h_95], title='ACF za AR(2) vremensku seriju'):
```

L8: Ključne kompetencije i literatura

Co-funded by the European Union

Kompetencije stečene kroz L8 (16h)

- Primeniti kompletni ML radni tok: EDA -> priprema -> trening -> evaluacija -> deployment
- Implementirati klasifikacione i regresione modele (RF, XGBoost, SVM, NN) u sklearn/keras
- Detektovati anomalije u SCADA podacima bez označenih primera (Isolation Forest, Autoencoder)
- Izgraditi LSTM modele za predviđanje procesnih varijabli i procenu RUL opreme
- Objasniti predikcije ML modela operaterima korišćenjem SHAP vrednosti
- Razumeti fizički informisane NN (PINN) i surogatne modele za SEVESO consequence analize

Ključna literatura

- Géron A.: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3. izd., O'Reilly)
- Brunton S., Kutz J.N.: Data-Driven Science and Engineering (Cambridge Univ. Press, 2022)
- Venkatasubramanian V.: Artificial Intelligence in Chemical Engineering (Elsevier, 2019)
- Raissi M. et al.: Physics-Informed Neural Networks (J. Computational Physics, 2019)

L8: Ključne formule

Co-funded by the European Union

Evaluacione metrike — ključne formule

- Precision = TP/(TP + FP) | Recall = TP/(TP + FN) | $F_1 = 2PR/(P + R)$
- AUC-ROC: površina ispod ROC krive — mera sposobnosti razlikovanja klasa
- MSE = $(1/n)\cdot\Sigma(y_i - \hat{y}_i)^2$ | RMSE = $\sqrt{MSE}$ | $R^2 = 1 - SS_{res}/SS_{tot}$

Regularizacija i optimizacija

- Ridge (L_2): min MSE + $\lambda\cdot\Sigma w_i^2$ | Lasso (L_1): min MSE + $\lambda\cdot\Sigma|w_i|$
- Gradient Descent: $w = w - \alpha\cdot\partial L/\partial w$ | Adam: adaptivna stopa za svaki w
- Sigmoid: $\sigma(z) = 1/(1 + e^{-z})$ | ReLU: max(0, z) | Softmax: $\exp(z_i)/\Sigma_j \exp(z_j)$

Ključni algoritmi i kada ih koristiti

- Random Forest: robustan, interpretabilan, dobar za početnu liniju — uvek pokušati prvo!
- XGBoost/LightGBM: kada je tačnost ključna, tabelarni podaci — state-of-the-art
- LSTM: sekvencionalni podaci, dugoročne zavisnosti, RUL predikcija
- Isolation Forest: detekcija anomalija bez labela — SCADA monitoring
- Autoencoder: kompleksna detekcija anomalija na visoko-korelisanim procesnim podacima
- PINN: kada ima malo podataka, ali poznate fizičke jednačine — nije crna kutija!

Preporučene Python biblioteke za praktične projekte

- scikit-learn, xgboost, lightgbm, keras, torch, shap, optuna, statsmodels, pyod
- pyod: Python Outlier Detection — 40+ metoda za detekciju anomalija u jednoj biblioteci